Building reproducible analytical pipelines with Python

Bruno Rodrigues

2024-01-13

Table of contents

Welcome!

How using a few ideas from software engineering can help data scientists, analysts and researchers write reliable code

This is the Python edition of the book "Building reproducible analytical pipelines", and it's a work-in-progress. If you're looking for the R version, visit this link.

Data scientists, statisticians, analysts, researchers, and many other professionals write a lot of code.

Not only do they write a lot of code, but they must also read and review a lot of code as well. They either work in teams and need to review each other's code, or need to be able to reproduce results from past projects, be it for peer review or auditing purposes. And yet, they never, or very rarely, get taught the tools and techniques that would make the process of writing, collaborating, reviewing and reproducing projects possible.

Which is truly unfortunate because software engineers face the same challenges and solved them decades ago.

The aim of this book is to teach you how to use some of the best practices from software engineering and DevOps to make your

projects robust, reliable and reproducible. It doesn't matter if you work alone, in a small or in a big team. It doesn't matter if your work gets (peer-)reviewed or audited: the techniques presented in this book will make your projects more reliable and save you a lot of frustration!

As someone whose primary job is analysing data, you might think that you are not a developer. It seems as if developers are these genius types that write extremely high-quality code and create these super useful packages. The truth is that you are a developer as well. It's just that your focus is on writing code for your purposes to get your analyses going instead of writing code for others. Or at least, that's what you think. Because in others, your team-mates are included. Reviewers and auditors are included. Any people that will read your code are included, and there will be people that will read your code. At the very least future you will read your code. By learning how to set up projects and write code in a way that future you will understand and not want to murder you, you will actually work towards improving the quality of your work, naturally.

The book can be read for free on https://b-rodrigues.github.io/raps_with_py/ and you'll be able buy a DRM-free Epub or PDF on Leanpub[1] once there's more content.

You can submit issues, PRs and ask questions on the book's Github repository[2].

[1]https://leanpub.com/

[2]https://github.com/b-rodrigues/raps_with_py

Preface

I don't like Python. Or rather, don't like using it to analyze data. I believe that certain design choices were made that make Python a subpar language for analyzing data. That being said, Python is currently the most popular general purpose programming language, and also the most popular language to analyze data, especially in machine learning and AI. As such, it is a good idea to at least know your way around it even if, like me, you prefer using the R programming language to analyze data.

I state that I don't like Python because I want to make something very clear, right from the start. I am not an expert in Python. I know enough to get things done, but it is not a language that I know well. I consider myself an expert in R, but definitely not in Python. So why write a book on Python in that case? Well, in truth, this is not a book about Python.

This book is about building reproducible pipelines. And in order to build reproducible pipelines, you need to apply certain general ideas. This book will discuss these ideas, and illustrate them using the Python programming language. Acutally, I wrote this book already: you can read it here. The book was well received: lots of people contacted me to thank me for having written it, I was invited to give talks several times on the book already, animate some workshops and some people are already asking me for a second edition.

So this is what I decided to teach the students: how they could structure their projects in such a way that they could spot problems like that during development, but also make it easy to reproduce and retrace who did what and when. I wrote my course notes into a freely available bookdown that I used for teaching. When I started compiling my notes, I discovered the concept *Reproducible Analytical Pipelines* as developed by the Office for National Statistics (henceforth ONS). I found the name "Reproducible Analytical Pipeline" (henceforth RAP) really perfect for what I was aiming at. The ONS team responsible for evangelising RAPs also published a free ebook in 2019 already. Another big source of inspiration is Software Carpentry to which I was exposed during my PhD years, around 2014-ish if memory serves. While working on a project with some German colleagues from the University of Bonn, the principal investigator made us work using these concepts to manage the project. I was really impressed by it, and these ideas and techniques stayed with me since then.

The bottom line is: the ideas I'm presenting here are nothing new. It's just that I took some time to compile them and make them accessible and interesting (at least I hope so) for users of the R programming language.

At least my students found the course interesting. But not just students. I tweeted about this course and shared the notes with a wider audience, and this is when I got very positive feedback from people that were not my students. People wanted to buy this as a book and go deeper into the topics laid out. This is when I realised that, as far as I know, there is not a practical book available discussing these topics. So I decided to write one, but I took my time getting started. What finally, really, got me working on it was when Dmytro Perepolkin reached out to me and suggested I contact several persons to get their inputs and

ideas and get started. I followed his advice, and this led to very fruitful discussions with Sébastien Rochette, Miles McBain and Dmytro. Their ideas and inputs definitely improved the quality of this book, so many thanks to them. Also thanks to David Solito, Matan Hakim, Stas Kolenikov, Sam Parmar, Chuck, Matouš Eibich, Jonathan Moore, Alain Vagner and Matthias Meiksner for proofreading the book and providing valuable feedback and fixes. And thank you, dear reader, for picking this up!

This book is divided into two parts. The first part teaches you what I believe is essential knowledge you should possess in order to write truly reproducible pipelines. This essential knowledge is constituted of:

- Version control with Git and how to manage projects with Github;
- Functional programming;
- Literate programming.

The main idea from part 1 is "don't repeat yourself". Git and Github will help us avoid losing code, and losing track of who should do what in a project (even if you're working alone on a project, you will see that using Git and Github will save you many hours and headaches). Getting familiar with functional and literate programming should improve the quality of our code by avoiding two common sources of mistakes: computing results that rely on the state of our program (and later, the state of the whole hardware we are using) and copy and paste mistakes.

The second part of the book will then build upon this knowledge to introduce several tools that will help us go beyond the benefits of version control and functional and literate programming:

- Dependency management with `{renv}`;
- Package development with `{fusen}`;

- Unit and assertive testing;
- Build automation with `{targets}`;
- Reproducible environments with Docker;
- Continuous integration and delivery.

While this is not a book for beginners (you really should be familiar with R before reading this), I will not assume that you have any knowledge of the tools presented in part 2. In fact, even if you're already familiar with Git, Github, functional programming and literate programming, I think that you will still learn something useful from reading part 1. But be warned, this book will require you to take the time to read it, and then type on your computer. Type *a lot*.

I hope that you will enjoy reading this book and applying the ideas in your day-to-day, ideas which hopefully should improve the reliability, traceability and reproducibility of your code. You can read this book for free on https://raps-with-r.dev/, or if you want you can buy a DRM-free PDF or Epub over at https://leanpub.com/raps-with-r.

If you want to get to know me better, read my bio[3].

If you have feedback, drop me an email at bruno [at] brodrigues [dot] co.

Enjoy!

[3]https://www.brodrigues.co/about/me/

References