The Open Science Manual

Make Your Scientific Research Accessible and Reproducible

Claudio Zandonella Callegher and Davide Massidda

18 April, 2022 [last-updated]

Contents

Preface

The present book aims to describe programming good practices and introduce common tools used in software development to guarantee the reproducibility of the analysis results. We want to make scientific research open-source knowledge development.

The book is available online at https://arca-dpss.github.io/manual-open-science/.

A PDF copy is available at [TODO: add link].

Book Content

In the book, we will learn to:

- Share our materials using the Open Science Framework (**OSF**)
- Learn how appropriately to structure and organize our materials in a **repository**
- Follow recommendations about data organization and data sharing
- Improve code readability and maintainability using a **Functional Style**
- Learn version control and collaboration using **Git** and **Github**
- Manage analysis workflow with dedicated tools
- Create dynamic documents
- Manage project requirements and dependencies using dedicated tools
- Create a container to guarantee reproducibility using **Docker**

This book provides useful recommendations and guidelines that can be applied independently of the specific programming language. However, examples and specific applications are based on the R programming language. Readers working with programming languages other than R can still find valuable guidelines and information and can later apply the same workflow and ideas using dedicated tools specific to their preferred programming language.

Finally, as most researchers have no formal training in programming and software development, we provide a very gentle introduction to many programming concepts and tools without assuming any previous knowledge. Note, however, that we assume that the reader is already familiar with the R programming language for specific examples and applications.

About the Authors

During our careers, we both moved into the field of Data Science after our PhD in Psychological Sciences. This book is our attempt to bring back into scientific research what we have learned outside of academia.

- Claudio Zandonella Callegher. During my PhD, I fell in love with data science. Understanding the complex phenomena that affect our lives by exploring data, formulating hypotheses, building models, and validating them. I find this whole process extremely challenging and motivating. Moreover, I am excited about new tools and solutions to enhance the replicability and transparency of scientific results.
- Davide Massidda.

ARCA

ARCA courses are advanced and highly applicable courses on modern tools for research in Psychology. They are organised by the Department of Developmental and Social Psychology at the University of Padua.

Contribute

Surely there are many typos to fix and new arguments to include. Anyone is welcome to contribute to this book. For small typos just send a pull request with all the corrections. To propose new chapters or paragraphs, instead, open an issue to discuss and plan them.

Acknowledgements

This book was inspired by Richard McElreath's talk *"Science as Amateur Software Development"* https://youtu.be/zwRdO9_GGhY. Talk abstract:

> Science is one of humanity's greatest inventions. Academia, on the other hand, is not. It is remarkable how successful science has been, given the often chaotic habits of scientists. In contrast to other fields, like say landscaping or software engineering, science as a profession is largely *unprofessional* - apprentice scientists are taught less about how to work responsibly than about how to earn promotions. This results in ubiquitous and costly errors. Software development has become indispensable to scientific work. I want to playfully ask how it can become even more useful by transferring some aspects of its professionalism, the day-to-day tracking and back-tracking and testing that is especially part of distributed, open-source software development. Science, after all, aspires to be distributed, open-source knowledge development.

License

1

Introduction

> Science is one of humanity's greatest inventions. Academia, on the other hand, is not. It is remarkable how successful science has been, given the often chaotic habits of scientists. In contrast to other fields, like say landscaping or software engineering, science as a profession is largely *unprofessional* - apprentice scientists are taught less about how to work responsibly than about how to earn promotions. This results in ubiquitous and costly errors. Software development has become indispensable to scientific work. I want to playfully ask how it can become even more useful by transferring some aspects of its professionalism, the day-to-day tracking and back-tracking and testing that is especially part of distributed, open-source software development. Science, after all, aspires to be distributed, open-source knowledge development.
>
> Richard McElreath's "Science as Amateur Software Development" talk
>
> https://youtu.be/zwRdO9_GGhY

Richard McElreath's words are as enlightening as always. Usually, researchers start their academic careers led by their great interest in a specific scientific area. They want to answer some specific research question, but these questions quickly turn into data, statistical analysis, and lines of code, hundreds of lines of code. Most researchers, however, receive essentially no training about programming and software development good practices resulting in very chaotic habits that can lead to costly errors. Moreover, bad practices may hinder the transparency and reproducibility of the analysis results.

Thanks to the Open Science movement, transparency and reproducibility are recognized as fundamental requirements of modern scientific research. In fact, openly sharing study materials and analyses code are prerequisites for allowing results replicability by new studies. Note the difference between replicability and reproducibility (Nosek & Errington, 2020):

- **Reproducibility**, obtaining the results reported in the original study using the *same data* and the *same analysis*.
- **Replicability**, obtaining the results reported in the original study using *new data* but the *same analysis* (a new study with the same experimental design).

So, reproducibility simply means re-running someone else's code on the same data to obtain the same result. At first, this may seem a very simple task, but actually, it requires properly organising and managing all the analysis material. Without adequate programming and software development skills, it is very difficult to guarantee the reproducibility of the analysis results.

The aim of the present book is to describe programming good practices and introduce common tools used in software development to guarantee reproducibility of the analysis results. Inspired by Richard McElreath's talk, we want to make scientific research an open-source knowledge development.

1.1 Book Structure

The book is structured as follows.

- In Chapter [TODO: add ref], we introduce the Open Science Framework (OSF), a free, open-source web application that allows researchers to collaborate, document, archive, share, and register research projects, materials, and data.
- In Chapter [TODO: add ref], we describe recommended practices to organize all the materials and files of our projects and which are the advantages of creating a well structured, documented, and licensed repository.
- In Chapter [TODO: add ref], we discuss the main guidelines regarding organizing, documenting, and sharing data.
- In Chapter [TODO: add ref], we provide general good practices to create readable and maintainable code and we describe the functional style approach.
- In Chapter [TODO: add ref], we introduce version control, a powerful system for managing the development of our project. In particular, first, we provide a basic tutorial about the use of the terminal. Next, we introduce Git and GitHub for managing and tracking our projects during the development.
- In Chapter [TODO: add ref], we discuss how to manage the analysis workflow to enhance results reproducibility and code maintainability.
- In Chapter [TODO: add ref], we introduce the main tools to create dynamic documents that integrate narrative text and code describing the advantages.

- In Chapter [TODO: add ref], we discuss how to manage our project requirements and dependencies (software and package versions) to enhance results reproducibility.
- In Chapter [TODO: add ref], we introduce Docker and the container technology that allows us to create and share an isolated, controlled, standardized environment for our project.

1.2 Instructions

Let's discuss some useful tips about how to get the best out of this book.

1.2.1 Programming Language

This book provides useful recommendations and guidelines that can be applied independently of the specific programming language used. However, examples and specific applications are based on the R programming languages.

In particular, each chapter first provides general recommendations and guidelines that apply to most programming languages. Subsequently, we discuss specific tools and applications available in R.

In this way, readers working with programming languages other than R can still find valuable guidelines and information and can later apply the same workflow and ideas using dedicated tools specific to their preferred programming language.

1.2.2 Long Journey

To guarantee results replicability and project maintainability, we need to follow all the guidelines and apply all the tools covered in this book. However, if we are not already familiar with all these arguments, it could be incredibly overwhelming at first.

Do not try to apply all guidelines and tools all at once. Our recommendation is to build our reproducible workflow gradually, introducing new guidelines and new tools step by step at any new project. In this way, we have the time to learn and familiarize ourselves with a specific part of the workflow before introducing a new step.

The book is structured to facilitate this process, as each chapter is an independent step to build our reproducible workflow:

- Share our materials using online repositories services
- Learn how to structure and organize our materials in a repository
- Follow recommendations about data organization and data sharing
- Improve code readability and maintainability using a Functional Style
- Learn version control and collaboration using Git and Github
- Manage analysis workflow with dedicated tools
- Create dynamic documents
- Manage project requirements and dependencies using dedicated tools

- Create a container to guarantee reproducibility using Docker

Learning advanced tools such as Git, pipeline tools, and Docker still requires a lot of time and practice. They may even seem excessive complex at first. However, we should consider them as an investment. As soon as our analyses will become more complex than a few lines e of code, these tools will allow us to safely develop and manage our project.

1.2.3 Non-Programmer Friendly

Most of the arguments discussed in this book are the A-B-C of the daily workflow of many programmers. The problem is that most researchers lack any kind of formal training in programming and software development.

The aim of the book is exactly that: to introduce popular tools and common guidelines of software development into scientific research. We try to provide a very gentle introduction to many programming concepts and tools without assuming any previous knowledge. Note, however, that we assume the reader is already familiar with the R programming language for specific examples and applications.

References

Nosek, B. A., & Errington, T. M. (2020). What is replication? *PLOS Biology*, *18*(3), e3000691. https://doi.org/10.1371/journal.pbio.3000691