# WJP - R Coding Handbook

Carlos A. Toruño P.

November, 2022

# Table of contents

**Introduction** **3**
Prerequisites . . . . . 4
Handbook Structure . . . . . 4
Licence . . . . . 5

**1 Workflow** **6**
1.1 SharePoint . . . . . 6
1.2 Git . . . . . 7
1.3 Projects . . . . . 8
1.4 Data Management . . . . . 8
1.4.1 File Organization . . . . . 8
1.4.2 Documentation . . . . . 9

**2 Coding** **11**
2.1 Script Headline and Outline . . . . . 11
2.2 Loading Packages . . . . . 13
2.3 Coding Style . . . . . 14
2.3.1 The Tidyverse Style guide . . . . . 14

# Introduction

Figure 0.1: Zacatal by nicaraguan painter Raúl Marín

This coding handbook is a continuous work maintained by the Data Analytics Unit (DAU) from The World Justice Project (WJP) with the aim of unifying the different aspect of the carried by the unit. In this book, you will find not only general guidelines but also several issues that will help the reader to understand and contribute in our tasks.

The handbook cover aspects related to the general workflow, the coding process and the visualization guidelines used by the team. As mentioned earlier, this book is a continuous work in progress and, as such, the rules and guides are not written in stone but rather, subject to improvements that every member of the team is open to discuss and include in this handbook.

## Prerequisites

In order to assimilate the content in this chapter (and the entire handbook) we will require you to have the following:

- An intermediate knowledge of R language.
- R Studio is installed in your computer, however, you are free to use any other Integrated Development Environment (IDE) or the Text Editor of your choice.
- A basic knowledge of Git.
- A GitHub account.
- Access to the DAU's SharePoint
- Although is not required, we strongly advise you to install GitHub Desktop.

If you are new to R, I would recommend you to check the Hands-On Programming with R book written by Garrett Grolemund. If you are already comfortable with the R programming language but you need an introduction on how to analyze data using it, I would suggest you to read the R for Data Science book edited by the R4DS team.

## Handbook Structure

The content in this Handbook is organized as follows:

- **Part I: Workflow**: A brief chapter in which we will cover the basic guidelines and issues related to the DAU workflow when programming with R and R Studio along with the basic setup.
- **Part II: Coding**: The main chapter of the handbook where we discuss the main aspects of the coding style when working in projects related to the DAU.
- **Part III: Viz Aesthetics**: Manual of style, style guidelines, pre-defined settings and other aspects related exclusively to data visualization are presented in this chapter.
- **Appendix**: Final part of the handbook where you can read about R-related content for the daily tasks that you will have to face as a member of the DAU.

## Licence



To learn more about the work of The World Justice Project, please visit the official website.

# 1 Workflow

In this chapter, we will cover the basic guidelines and issues related to the prefered workflow when programming with R programming language. This chapter will cover four different aspects: a) *Cloud Storage*, b) *Version Control System*, c) *R Studio Projects*, and d) *File Management practices*.

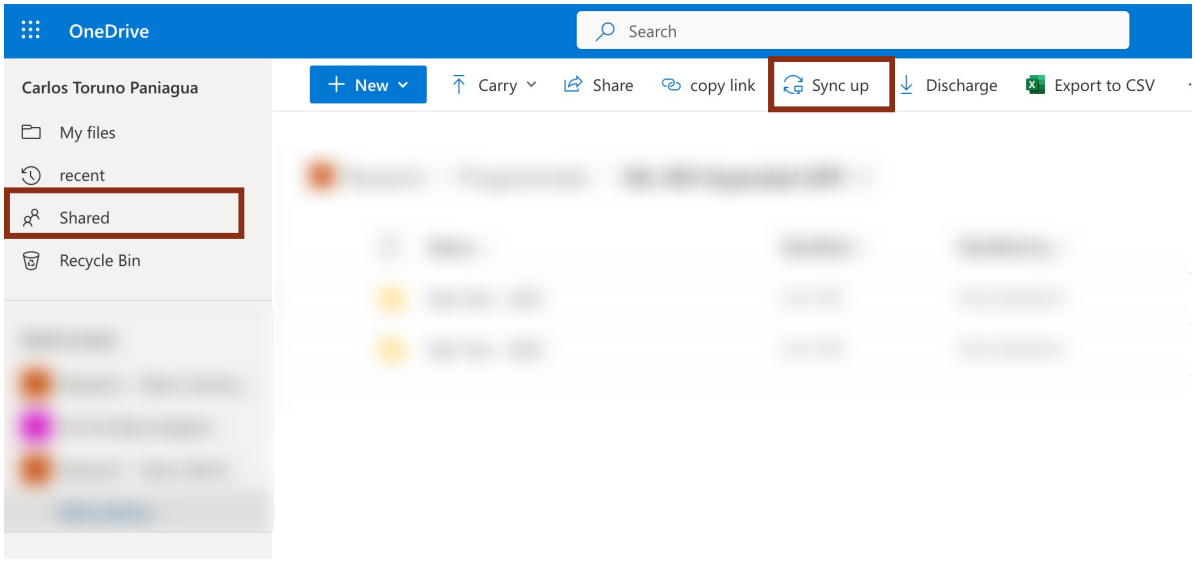
## 1.1 SharePoint

All files used by the DAU are stored in the Data Analytics folder within the organization's SharePoint. This is done in order to provide an easy access to all members of the team. As a rule, you are required to work and modify these files directly from this online directory. To achieve this, you will need to download the OneDrive app in your computer and sign in using your WJP account. If you are already using One Drive with your personal account, you can add an additional professional account which will function parallel to your personal account in your computer. For more information, see the following website.

Given that the DAU SharePoint is a directory created and administered by the organization, you will have to sync this directory to your own OneDrive WJP Account. In order to achieve this, follow these steps:

1. Login into your WJP Outlook account in any web browser.
2. Once into your account, access ***OneDrive***. This will open a new tab for you.
3. In the left side panel, locate the ***Share with Me***tab and click it.
4. Locate the ***Data Analytics folder***.
5. Once that you are inside the folder, you will see the ***Add Shortcut to My Files***,or, ***Sync Up*** option in the upper toolbar, depending on your OneDrive version.
6. Follow the instructions and sync it to your WJP folder in your computer.

By syncing your work with the SharePoint, the whole team is going to be able to see and review changes as they go. However, if more than one person is working on the same file at the same time, it is likely that this working flow will produce several mistakes. Therefore, additionally to this shared cloud storage system, we need a version control tool that will allow us to modify and collaborate in team projects without these kind of issues. Given that most of our work is focused in coding, we use ***Git Version Control*** and the ***GitHub*** platform for this.




## 1.2 Git

Git is a free and open source software that allows users to set up a version control system designed to handle projects. Given the nature of its features, it is normally used for collaboratively developing code and data integrity. GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. For a gentle introduction to Git and GitHub, see the following post published by Kinsta. For a more in depth introduction, please refer to the GitHub documentation or watch this video tutorial.

Using the Git features allow us to simultaneously work on the same project and even in the same code without worrying about interfering with other members of the team. As a rule, every project carried by the DAU has a code administrator who is in charge of setting up the GitHub repository and add other members of the team as project collaborators. Additionally, the code administrator is in charge of setting the *main branch* and the initial structure of the code (see the (**data-management?**) section on this chapter). It is required that the GitHub repository have its *main branch* in its respective SharePoint folder.

**Note:** It is highly recommended to create the repository from GitHub.com and not from the local machine to avoid the initial commit that can include system files such as **.DS_STORE** files

We use convergence development[1] to collaboratively code in the same project. For this, it is highly recommended that each team member works on a separate *branch* and, once the data routine is done, the auxiliary branches can be merged into the main branch of the repository.

[1]For more information about convergence development and branches, we advise you to refer to this article from Pluralsight.

Collaborators that are not the code administrator have the option to clone the GitHub repository in their computer in a local directory that it is not sync to the SharePoint and work in their respective branch from a local copy outside the SharePoint. In other words, it is only the functional final version contained inside the *main branch* the one that it is going to be sync in SharePoint.

**Important:** GitHub is used to keep track of the code we use in each project. Under no circumstance, we will include the data sources in the online repositories.

## 1.3 Projects

When you load a data set or source an R script, you will have to set up the working directory where these files are located in your computer. However, the path to this working directory is quite different for all the members of the team. R Studio allow us to enclose all of our analysis, code and auxiliary files into a project.

A project is a feature that allow us to work with the analysis we are carrying without having to worry about where does these files are stored or who is working on them. Say goodbye to `setwd("...")`. Besides managing relatives paths, R projects allow users to keep a history of actions performed and even keep the objects in your environment. Because of this, projects are the cornerstone of our work when performing analysis with R.

As a rule, every project has a file named `project-name.Rproj` in its root directory and open it should be the first action when working on a project. For more information on working with R projects, refer to the Workflow section from the R for Data Science book.

## 1.4 Data Management

The University of California San Diego (UC San Diego) has a Data Management Best Practices that reviews common guidelines for managing research data. In this handbook, I will focus on two topics mentioned by those guidelines: File Organization and Documentation.

### 1.4.1 File Organization

The file organization involves two important elements: filing system and naming conventions.

A filing system is basically the organization structure (directories, folders, sub-folders, etc) in which files are stored. There are no standard rules about how this should be done. However, the chosen filing system needs to make sense not only to the person currently working on a given project but to anyone going through these files in the future. As a rule, each project would have the following sub-folders:

- Code: Depending on the complexity of the project, you could choose to create separate directories within the Code folder for Stata, R or Python files.
- Data: Depending on the complexity and nature of the project, you could choose to create separate directories within the Data folder for RAW, INTERMEDIATE or CLEAN data sets.
- Outcomes: The outcomes of the project might have several different formats. For example, images could be in PNG and/or SVG format, Reports might be created in PDF, some tables might be exported as TEX files, etc. The outcomes folder should have a separate directory for each one of these formats.

In some cases, the creation of a PDF report is key, for example, the Regional Country Reports. For these projects, we strongly advise to create a separate directory to store the code files used for the report. At the moment of writing this handbook, these reports are created using R Markdown, but a migration to Quarto is feasible in the future.

- Markdown/Quarto

In relation to the naming conventions, these are a set of rules designed to complement the filing system and help collaborators in understanding the data organization. Each project have the flexibility to use a specific set of naming rules to use in the filing system. However, there are a few general rules to note:

- Use descriptive file names that are meaningful to you and your colleagues while also keeping them short.
- Avoid using spaces and make use of hyphens, snake_case, and/or camelCase.
- Avoid special characters such as $, #, ^, & in the file names.
- Be consistent not only along the project but also across different projects. If all different data files and routines are named in the same way, it's easier for you to use those tools across projects and re-factor routines.

### 1.4.2 Documentation

When initializing the GitHub repository, we strongly suggest to include a README file with it. Such file should be a Markdown document including the following elements:

- A brief description of the project and the role of the DAU in it.
- Referred person contact, which should the project leader and the code administrator.
- A brief description of the filing system along with what to find in each sub-directory.

- Given that no data is uploaded into the online repositories, include the following descriptions:
  - Data needed to run the code: source and process to obtain it.
  - Location of the data in the SharePoint.
- A brief description on how to read the code with, if possible, visual aids.

For an example on one of these files, please check the following README file. If possible, use this example as a template for future projects.

# 2 Coding

In this chapter, we will cover some basic guidelines and styling rules related to how the coding is done at the DAU-WJP. When programming data analysis routines, it is very hard to vanquish the personal style that every person has. Therefore, this chapter is focused on giving general guidelines that will allow to standardize their codes for an easy collaboration among all team members.

Remember, when writing a code, you are just the author not the audience. Therefore, think on how would other people understand what you are writing without the ideas and knowledge you have at the moment of creating your code. Writing comments, using titles as step-by-step guides, documenting the issues, all of these actions will greatly help other team members to understand what you have done and why you have done it. Also, this will help you in the future to understand what you did in the past and reduce the level of dependency of a given project on its collaborators.

## 2.1 Script Headline and Outline

The headline is very important because it gives the general information about the script, its purposes, the authors, the program version, among other important details. Within the DAU, we have the following template:

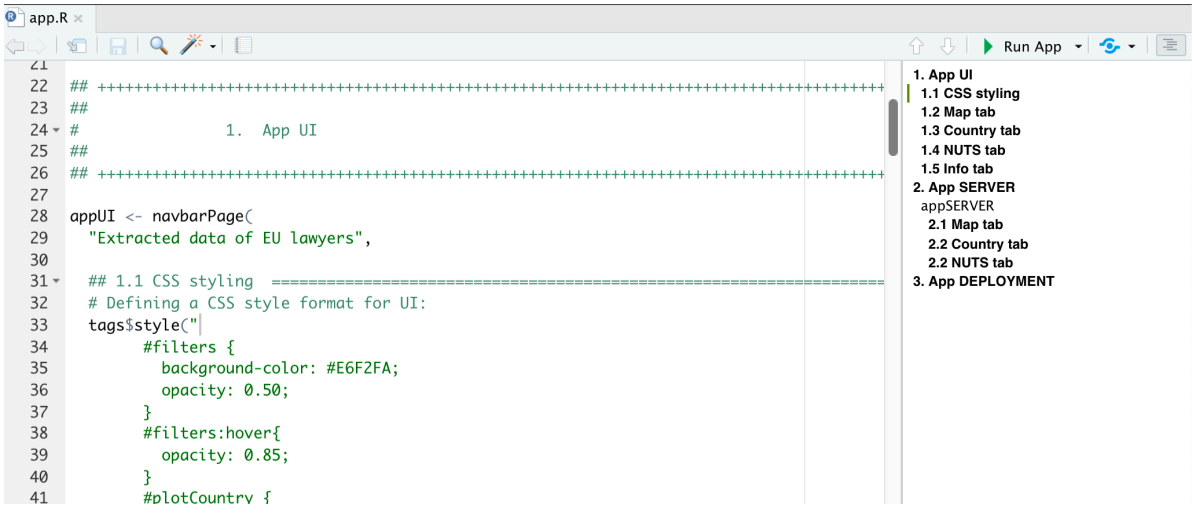
```
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
##
## Script:            PROJECT NAME - Script Purpose
##
## Author:            Author 1 Name   (email)
##                    Author 2 Name   (email)
##
## Creation date:     Month Day, Year
##
## This version:      Month Day, Year
##
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

As you can see, the template highlights the most important information that we need to know when opening a script. This information should give any team member a general idea on the project status even if this person has never collaborated in the project before. However, this is just a first step in the process. Our template goes beyond and it also displays how to use Titles, Subtitles and Steps within the script:

```
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
##
#                 1.  TITLE
##
## ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


## 1.1 SUBTITLE  ================================================================================

# Step 1

# Step 2

## 1.2 SUBTITLE  ================================================================================

# Step 1

# Step 2
```

Unlike the Headline seen above, the use of this outline to structure your code will vary from script to script. For example, if you are working on a very short routine of less than 100 lines. You might not need to use titles and subtitles and you would rather choose to only use steps. Therefore, the extent to which this outline is feasible will be a decision of the code administrator.

The main objective of this outline is to provide an organized outline for the script. The utility of this outline increase with the complexity of the code. Although we strongly advice you to avoid very long routines (we will come back to this when we talk about refactoring and modules), we understand that sometimes the code might be very complex by nature. As an example, look at the following script from the Shiny App that we prepared for the EU Project. Even though the code extends for less than 500 lines, it is difficult to read due to the ***reactivity*** and ***laziness*** concepts that come along when programming a Shiny App.




## 2.2 Loading Packages

On any routine, the first thing is to load the packages that you will be using in the code. Usually, you will see routines calling the `library` or `request` modules to load the needed packages. However, given that this is a team collaboration, not everyone have these packages installed or, even if they have it, the script requires a certain version of the package in order to run. Therefore, we need to, not only load, but also check for these other requirements as well.

W=To achieve this, we could write down a series of if conditions. However, we choose to rely on the pacman package to do this. As such, we only request that every team member has this package installed in their local machine. If every team member has it installed, it is quite easy to load the required libraries by calling the `p_load` function. Take a look at the following example:

```
# Required packages
library(pacman)

# Development version
p_load_gh(char = c(

  # Visualizations
  "xl0418/ggradar2", "davidsjoberg/ggsankey",

  dependencies = T
))
```

```
# Stable CRAN release
p_load(char = c(

  # Visualizations
  "showtext", "ggtext", "ggsankey", "ggrepel", "ggplotify", "gridExtra", "ggradar2",
  "patchwork",

  # Data Loading
  "haven", "readxl",

  # Other
  "margins",

  # Good 'ol Tidyverse
  "tidyverse"

))
```

Three things are worth noticing. First, if some packages require the development version, we use the `p_load_gh` to install their latest release from GitHub. Second, we always install the development releases first and then the stable CRAN version at last. This is done in order to leave the tidyverse for last and avoid any other package to mask over the tidyverse functions. Third, for very complex routines, we might need to load several packages, it is strongly recommended to comment on the utility and need for each package so all team members can have a general idea on their use before reading the respective documentation.

## 2.3 Coding Style

### 2.3.1 The Tidyverse Style guide

In the DAU, we strongly rely on the guidelines defined by *Hadley Wickham* in its Tidyverse Style Guide. We request that every DAU member read this style guide before collaborating with other team members in a given project. As stated in the first page of the guide:

> *Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread.*

This guide is, as its name suggest, just a guide. As such, some of its guidelines might not be consistent with the coding style that the DAU as a whole have. Therefore, we do not rely on the use of styler and lintr packages to style our codes. In what follows, we will highlight

some topics and also complement with some of our own style guidelines the aforementioned reference.