Facultad
de
Ciencias

Libro Ejecutable Para el Atlas del Cambio Climático

(Executable Book for Climate Atlas)

Trabajo de Fin de Grado
para acceder al

GRADO EN FÍSICA

Autor: David Domínguez Román

Director: Antonio S. Cofiño

Junio-2023

Palabras Clave: JupyterBook, Interactive Atlas, IPCC, Cambio Climático, Libro Ejecutable, Binder

Keywords: JupyterBook, Interactive Atlas, IPCC, Climate Change, Executable Book, Binder

Resumen: Se ha creado un libro computacional mediante el ecosistema *JupyterBook*, con los scripts de código y los notebooks del *Multi-Model Intercomparison Project (Multi-MIP) Climate Change Atlas repository* para mejorar su reproducibilidad y la reutilización de su código.

El *Multi-MIP Climate Change Atlas repository* es la columna vertebral del *Sixth Assesment Report (AR6)* del *Grupo Intergubernamental de expertos sobre el Cambio Climático (IPCC)*, el cual proporciona una evaluación, región por región, del cambio climático, incluyendo el innovador Atlas Interactivo. El *AR6* fomenta los principios *Encontrable, Accesible, Interoperable y Reutilizable (FAIR)* para los datos científicos.

Abstract: A computational book has been created using the *JupyterBook* ecosystem, with the code scripts and the notebooks from the *Multi-Model Intercomparison Project (Multi-MIP) Climate Change Atlas repository* to improve its reproducibility and reusability.

The *Multi-MIP Climate Change Atlas repository* is the backbone of the *Sixth Assesment Report (AR6)* from the *Intergovernmental Panel on Climate Change (IPCC)*, that provides a region-by-region assessment of climate change, including also the innovative Interactive Atlas. The *AR6* promotes the Findable, Accessible, Interoperable and Reusable (FAIR) principles for scientific data.

Executable Book for Climate Change Atlas

David Domínguez Román

June 2023

Abstract

A computational book has been created using the *JupyterBook* ecosystem, with the code scripts and the notebooks from the *Multi-Model Intercomparison Project (Multi-MIP) Climate Change Atlas repository* to improve its reproducibility and reusability.

The *Multi-MIP Climate Change Atlas repository* is the backbone of the *Sixth Assesment Report (AR6)* from the *Intergovernmental Panel on Climate Change (IPCC)*, that provides a region-by-region assessment of climate change, including also the innovative Interactive Atlas. The *AR6* promotes the Findable, Accessible, Interoperable and Reusable (FAIR) principles for scientific data.

Contents

1 Introduction

The process of an investigation nowadays tends to use a lot of computational power. Scientists create computational laboratories in order to process the large amount of data collected. The fact of using *Big Data* analysis makes very difficult to track and replicate the investigations. Once an investigation group publishes an article with the results, accessing the code used to process the data it is somewhat difficult, not just to get it but to build an environment to run it. In this context, the use of interactive books comes to solve it, providing everyone the access of all code from the investigation in real time without leaving the browser, letting them to run it, interact with it and get the results in real time.

This interactive books are not just meant to be for professional use. It can be used to make a virtual laboratory in case any problem occurs and going to a physical laboratory in high school or university become impossible. It can also be used to make a book for students to have the theory and a way to verify that the problems they are solving while studying are going as expected. Or maybe just a way of letting the students to play and have fun with the subject they are studying. It also can be used to make a book that shows in real time, and not by screen captures, how code is structured, and what each line of code is doing. The use of interactive books goes as far as the imagination of which creates them.

This Project comes as a way to allow everyone to use the code used to make the *IPCC WGI Interactive Atlas* (*Atlas*) possible, to help other scientist learn how the data processing of the records of temperature and precipitation has been made, how everything comes together and to provide the scientific community a way to contact with the original creators of the code, suggesting changes, reporting bugs,... and thus contributing to a book that can be continuously edited, instead of waiting years for a new edition of the book. New algorithms can be implemented, data from future years can be added,... It makes this Project a book that is always up to date.

1.1 Climate Change. The IPCC Atlas

Since the early 18Th century, scientists have discussed the global warming and the climate change. First, they investigated the different geological changes and how climate changes could have induced the *Ice Ages*. The first calculations of the *Greenhouse effect* were developed by *Svante Arrhenius* and *Tomas Chowder Chamberlain*, among others. The first person to use the term "greenhouse effect" was *Nils Gustaf Ekholm* in 1901.(*History of Climate Change Science - Wikipedia* 2023)

By the 1950s, there was and increasing concern about the effects that fossil fuels could make in the global climate. It was discovered that CO_2 and water vapor (more substances were studied later, like methane) could lead to a warm-up if the concentration in the atmosphere increased without control. *Syukuru Manabe* and *Richard Wetheral* developed the first computational model of climate change, modeling how different concentrations of CO_2 change the temperatures of the different layers of the atmosphere (Manabe and T. Wetherald 2023). For that, *Manabe* was awarded a share of the *Nobel Prized of Physics* in 2021.

The use of computers to analyse the data from temperatures and precipitations and create atmospheric models have been the most useful tool to understand the global warming and design solutions to palliate the effects of the increase in temperature. In 1988 the Intergovernmental Panel on Climate Change *(*IPCC*)* was created as a way to assess the science of climate change and to facilitate measures for governments to try to alleviate the rise in temperatures and every effect of global warming.(*About — IPCC* 2023)

One of the several projects that the *IPCC* published is the *Sixth Assessment Report (AR6)* (*Sixth Assessment Report — IPCC* 2023). This report promotes best practises in traceability and reproducibility of the results shown in the report, including the adoption of the *Findable, Accessible, Interoperable, and Reusable (FAIR)* principles for scientific data. One part of that report is the Atlas (*IPCC WGI Interactive Atlas* 2023), that shows the observed and projected precipitations and temperature in the entire globe. The process of creating the *Atlas* begun with some notebooks that explained the different scripts used to manage the data from the observations. Those notebooks are the base of this Physics Degree Project.

1.2 Reproducibility and Access to Inform

The democratization of the science has been one of the most important events that have occur to the humanity in terms of science advance. In most countries there is people constantly investigating and contributing to science. This comes with an important drawback: the traceability of the investigations become more and more difficult. Furthermore, the use of big data is the order of the day: large amounts of data are collected and processed for each investigation, making reproducibility a hard task. The possibility to unify the code used in the investigations and the text of the reports could lead to a better and easier way to reproduce and audit publications.

This problem not only affects the scientific community, it complicates people to access truthful information, making fake news and disinformation a real human disease. Scientists have a duty towards the population: to inform accurately, but at the same time in an easy and comprehensive way.

1.3 JupyterBook

JupyterBook *(JB)* is a software ecosystem based on Sphinx Engine *(SE)* for building beautiful publication-quality books and documents from computational material (The Jupyter Book Community 2020). It can read *Jupyter Notebooks* (`.ipynb` files), files written in *Markdown* (`.md`) with *MyST Markdown* syntax, *BiBTex* files (`.bib`) and URLs. It combines everything into a digital interactive book, enabling the creator to structure files in a coherent manner, to have a centralized system of cross references and bibliography and, more important, to have a continuous released project that can be updated in real time.

To organize everything, several configuration files have to be added to the Project. This files comes as a YAML file (`.yml`) with the required structure, packages needed for the project to be created, rules for the compile process, configuration of the bibliography, add-ons, etc. JB can, among others, run the code inside the `.ipynb` files, create a cache to avoid compiling unchanged files or implement different interactive environments such as *Binder* or *Google Collab*. The base of a *JupyterBook* are the *Jupyter notebooks*, files that can contain code and enrich text. They allow the user to see the code written inside, the output of the code, draw graphs, make tables and include rich media.

ForJB to create a project book with code inside, it is mandatory to create

a computational environment with all the software dependencies needed to reproduce every part of the code inside, not just to build the project book, but also to interact with it. For this task, *Conda* is the current standard approach to proceed, based on a YAML software specification file, which includes every dependency needed, so the reader can reproduce the execution of code in the project book locally. A better explanation of the *Conda* package manager will be introduce later.

1.4 Tools used to build the book

To build the book, several tools were needed (JB was explained before, so it does not nedd more explanation):

- *Jupyter Lab*: is a program included in the JupyterBook ecosystem that runs locally in the Conda environment, showing itself as a web page in the browser. It allows the interaction with the code inside the Project and can be used as a platform to edit the Jupyter notebooks and render them as input for the book building process.

- Conda: is a package and virtual environment software management system. It can manage software packages dependencies, including many programming languages. Allows the user to have different versions of software in different compartmentalized environments that can be updated and use independently. Conda will be the basic tool to re-create the computational environment required by the JB project.(*Conda — Conda Documentation* 2023)

- Zotero: is a program that allows the user to add bibliography to a list that can be exported to a file. It has cloud synchronization, plugins for every browser to automatically add a bibliographic reference to the system, a system to edit the different fields of each bibliographic reference (year, author, etc.), allows the user to change the nickname name of every reference and enable the user to share the bibliography with other users. To ease its use, the *Better BibTex* plugin for Zotero was used.(*Zotero | Your Personal Research Assistant* 2023)

- *Visual Studio Code*: is an IDE that was used to edit the files of the book. It has several extensions for Jupyter Books, R language, Python language, etc. It also has synchronization with *GitHub* and can manage

branches and repositories from it.(*Visual Studio Code - Code Editing. Redefined* 2023)

- *GitHub*: is a cloud platform for managing *Git* repositories with several extra features added. Every repository is divided in *branches*, sub-repositories that allows the user to work and improve the code written, without changing the current version that is accessible to everyone. It can be customized to automatized the compiling of the code, the change of files from one branch to another, the management of the issues created by the community and much more. One feature of *GitHub* that is essential for this book is the use of what is called *GitHub Pages*, a feature for implementing static content using HTML, CSS and JavaScript from the files inside the repository.(*GitHub* 2023)

- *Hypothesis*: is a plugin for the book deployed from *GitHub* that enables annotations and marking. It is a tool design for the community, to share annotations with everyone, although it can also be used by the creator of the book.(dwhly 2023)

- *Utterances*: is a plugin used as a way to comment the book and manage issues of *GitHub*. If any reader finds a bug or something wrong in the book, he can create an issue commenting the exact page in with the error is.(*Utterances* 2023)

- *Fedora Workstation*: is a *Linux* based operating system that was used because some of the packages needed to run the code from the book are not design to work with *Windows* systems. Fedora was selected due to the familiarity of the creator with the operating system, but every other Linux based system can be used, including the *WSL (Windows Sub-system for Linux)*.

- *Binder*: is a platform that will allow the user to execute the book and interact with it from the browser. For its use a Docker image is needed. Binder will run the docker image and will present in the browser an interface that is similar to that of the Jupyter Hub.(*The Binder Project* 2023)

- *Docker*: is a platform for the deploy of containerized applications. It is mandatory in order for the Binder to work.(*Docker: Accelerated, Containerized Application Development* 2023)

2 Process of Creating the Project Book

At first, it was important to become familiarized with the JB. The "Create your first Book" tutorial was the cornerstone of the Project, as it has the basic knowledge to create, structure, write, build and publish the book Project(*Create Your First Book* 2023).

The files used to create the Project come from the GitHub repository of IPCC-WG1/Atlas (*IPCC-WG1/Atlas: Repository Supporting the Implementation of FAIR Principles in the IPCC-WGI Atlas* 2023). This repository comes as a series of folders containing scripts, data and the `.ipynb` files. Every folder comes with a `README.md` file explaining the different steps that scripts make to generate what will be used by the notebooks. Figure 1 shows a basic scheme of how the Project works.

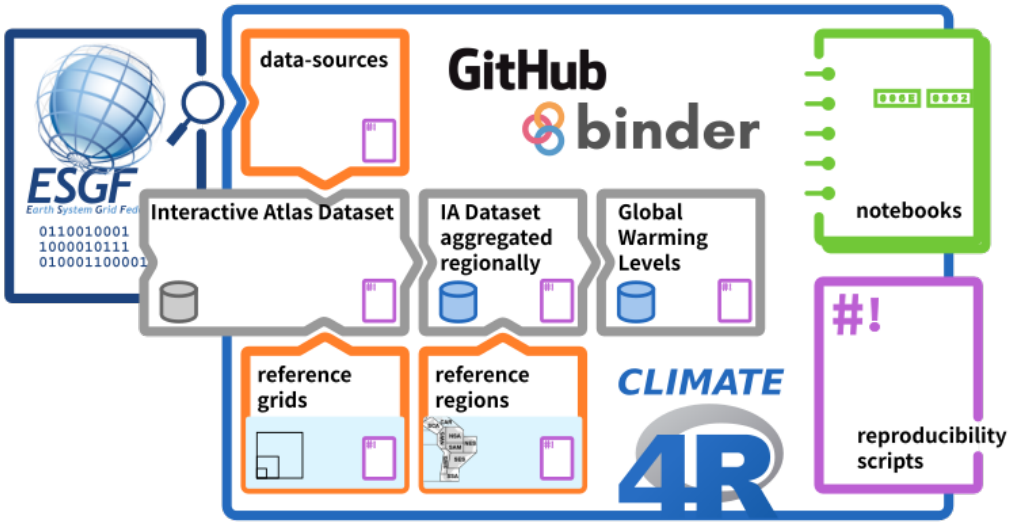


Figure 1: Scheme of how the Project book works (Iturbide et al. 2021).

```
Atlas/
    -binder/
    -datasets-aggregated-regionally/
    -datasets-interactive-atlas/
    -data-sources/
    -notebooks/
    -reference-grids/
    -reference-regions/
    -reproducibility/
    -warming-levels/
    -jb-cache/
    -Atlas-repo-scheme.svg
    -ERRATA.md
    -intro.md
    -LICENSE.md
    -logo.png
    -references.bib
    -_config.yml
    -_toc.yml
```

Listing 1: Structure of the Project.

The Project has several directories containing the data, scripts and the files explaining how everything works. There are also files with the licensing, the logo, an intro, etc. The `jb-cache` directory, and the files `_config.yml` and `_toc.yml` are the essential parts of the Project. They will be explained later (see section 2.1).

2.1 Configuring Jupyter Book

A project in JB starts with the creation of a template, a folder with all the basic files and configurations. This folder will be the root, the reference from which a path will be written. Instead of writing a path like `$HOME/Documents/MyProject/folder1/...`, the path will start from the folder of the project: `folder1/...`. The most important files in any JB project are `_config.yml` and `_toc.yml`.

The first one is the configuration file, a file that contains the basic metadata of the project, such as title, author, a logo, a path for the file with all the bibliography, several parameters regarding the process of building the project, the URL (if any), how JB will act if it encounters a file that it is not in the table of contents, set a timeout for the building process, etc. It can also manage a cache for every file used to build the project, to avoid compiling unchanged files and saving time. Listing 2 shows the different configurations for each part of the building process.

```yaml
title: Title of the project
author: Author(s) of the project
logo: logo from the organization
# Auto-exclude files not in the toc
only_build_toc_files: true
execute:
  execute_notebooks: how they will be executed
  # force: it executes every file allways
  # cache: saves the executed files and only
  #execute a file again if it changes.
  cache: path for the cache, if needed
  timeout: -1
# Define the name of the latex and PDF output
latex:
  latex_documents:
    targetname: name-of-document.tex
# Add a bibtex file
bibtex_bibfiles:
  - references.bib
# bib style
sphinx:
  config:
    bibtex_reference_style: several styles
    # author-year, alphabetic, ieee,...
    html_js_files: #to use things like Plotly
    - url of each file needed
# Where the book exists on the web
repository:
  url: Online location of the project
  path_to_book: Optional path to your book
  #relative to the repository root
  branch: Which branch of the repository
# Adding the launch buttons
launch_buttons:
  notebook_interface: what interface to use
  binderhub_url: if needed
  thebe: toggle thebe (T/F)
html:
  use_edit_page_button: toggle edit (T/F)
  use_issues_button: toggle issues (T/F)
  use_repository_button: toggle repository (T/F)
  comments: #tools for the community
    utterances:
      repo: user/name of the repo
    hypothesis: toggle hypothesis (T/F)
```

Listing 2: Example of a `_config.yml`

Listing 3 shows the actual configuration file of the Project. The first three entries are the title, author and a logo corresponding to the entity responsible for the Project. Then there is an instruction telling jupyter to only build files contained in the table of contents.

After that, there are configurations for the execution of the JB, such as if the Project is cached, where cache is placed and a timeout for the building process. In this case is set to "-1", meaning that there is no timeout limit. The next part is about LaTeX. JB can export the final Project to a `.tex` file and a `.pdf` file, and this parameter tells jupyter how it will be named.

The next two parts are related to the bibliography: one being the path to the `.bib` file containing the bibliography, and the other specifying the reference style.

Then there is information about the repository containing the Project and some options for the interactive part of the Project. After that, the interactive button to launch *Binder* and the *Thebe* button for using the live code environment, a mix between the text and the binder environment. Those buttons will work in the web version of the Project, both locally and through an URL pointing to the repository (see 2.6 and 2.5).

And finally, there are some settings regarding the web content and style (i.e. HTML and CSS). In this part, several configurations about the layout of the page are set, and the configuration of *Hypothesis* and *Utterances* are placed (see 2.5). There is the possibility to configure interactive notebook widgets as plots created using software like *Plotly* (*Plotly* 2023), *Bokeh* (Ven 2023) or *Jupyter Widgets* (*Jupyter Widgets — Jupyter Widgets 8.0.5 Documentation* 2023), though in this Project there are not any interactive plots.

```yaml
title: IPCC-ATLAS Jupyter Book
author: David Dominguez Roman
logo: logo.png
only_build_toc_files        : true
execute:
  execute_notebooks: cache
  cache: jb-cache/
  timeout: -1
latex:
  latex_documents:
    targetname: book.tex
bibtex_bibfiles:
  - references.bib
sphinx:
  config:
    bibtex_reference_style: author_year
repository:
  url: https://github.com/PhantomAurelia/Atlas
  path_to_book: .
  branch: jb-test
launch_buttons:
  notebook_interface: "jupyterlab"
  binderhub_url: "https://mybinder.org"
  thebe                   : true
html:
  use_edit_page_button: true
  use_issues_button: true
  use_repository_button: true
  comments:
    utterances:
      repo: "PhantomAurelia/Atlas"
    hypothesis: true
```

Listing 3: `_config.yml` of the Project.

2.2 Adding content and structure to Jupyter Book

The `_toc.yml` file refers the structure and content of the project. A `jb-book` project can be structured in parts, chapters and sections; but if everything in a continuous layout is what is needed, `jb-article` should be used. Listing 4 shows an example of this file. Each part can be numbered using the `numbered: True` parameter and named declaring the `caption:` parameter before declaring a chapter start. Each file, glob or URL can also named using the `title:` parameter. Every file contained in the `_toc.yml` has to be referred with a path relative to where the `_toc.yml` is.

```yaml
format: jb-book #jb-article can be used
root: intro #path to the file that will be
#used as the first page of the book
parts:
  - caption: name of part 1
    numbered: True #this part will be numbered
    chapters:
    - file: path relative to _toc.yml
      title: Alternative title
  - caption: name of part 2
    #no numbered option,
    #so this part will not be numbered
    chapters:
    - file: #several types of files can be added
    - file: file.md #in markdown
    - file: anotherfile.md #in MyST Markdown
      sections: #different sections in a chapter
      - file: file.ipynb #jupyter notebook
      - file: custom notebook format
      - glob: /folder1/../
            #variable that search
          #for multiple files in a specific path
      - url: external web page
            #starting with http or https
        title: Alternative title for the web page
```

Listing 4: Example of a `_toc.yml`

Figure 5 shows the current table of contents. This Project is formatted as a `jb-book`, with the root file being `intro.md`. There is no part numbered because it was considered not necessary. The Project is divided in three parts: Information, Notebooks and Appendix. Information has every file that was previously a `README.md` inside the folders of every code, explaining with detail how the Project works. Notebooks implement the `.ipynb` used for the management of the data to create the *Atlas*. Appendix includes the

`README.md` files of the scripts regarding reproducibility, auxiliary material, scripts for bash interpolation and information about binder.

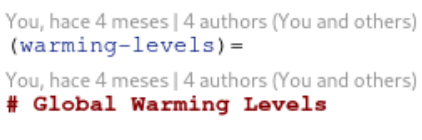
```
#Important: some paths have been divided in two or more lines,
#but to fit in the text width.
#In the original file paths go in one line without spaces.
format: jb-book
root: intro
parts:
  - caption: Information
    chapters:
      - file: warming-levels/warming-levels
      - file: data-sources/data-sources
      - file: reference-grids/reference-grids
      - file: datasets-interactive-atlas/datasets-interactive-atlas
      - file: reference-regions/reference-regions
      - file: datasets-aggregated-regionally/datasets-aggregated-regionally
      - file: notebooks/notebooks
  - caption: Notebooks
    chapters:
      - file: notebooks/regional-scatter-plots_R.ipynb
      - file: notebooks/global-warming-levels_R.ipynb
      - file: notebooks/bias-adjustment_R.ipynb
      - file: notebooks/CORDEX-overlaps_Python.ipynb
      - file: notebooks/GeoTIFF-post-processing_R.ipynb
      - file: notebooks/hatching-uncertainty_R.ipynb
      - file: notebooks/linear-trends_R.ipynb
      - file: notebooks/reference-grids_R.ipynb
      - file: notebooks/reference-regions_Python.ipynb
      - file: notebooks/reference-regions_R.ipynb
      - file: notebooks/stripes-plots_R.ipynb
  - caption: Appendix
    chapters:
      - file: ERRATA
      - file: reproducibility/reproducibility
      - file: notebooks/auxiliary-material/auxiliary-material
      - file: binder/binder
      - file: reproducibility/maps/maps
      - file: reproducibility/observations/observations
      - file: reproducibility/projections/projections
      - file: datasets-interactive-atlas/bash-interpolation-scripts/
                bash-interpolation-scripts
      - file: datasets-interactive-atlas/bash-interpolation-scripts/
                AtlasCDOremappeR_CORDEX/atlasCDOremappeR
```

Listing 5: `_toc.yml` file of the project

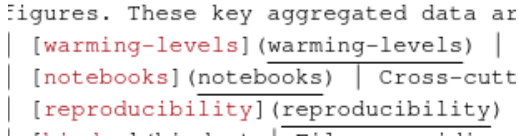
2.3 References in the Project

The text of this book Project uses three classes of references: cross references, bibliographic cites and download references. This references are *roles* of *Sphinx Engine*, but with a different syntax. A *role* in SE is an interpreted text that is used to insert semantic markups in the text (*Roles — Sphinx Documentation* 2023). How every class is used in the Project will be described below.

Cross references are used to refer to specific parts of the Project. To do that, a label with the `(desired label)=` structure has to be placed wherever it is wanted. If it is placed at the start, it will label the entire document file. Where it is needed to refer from, a reference with `[text shown](label)` structure must be written.



(a) `warming-level.md` label.



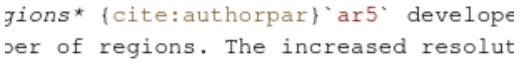
(b) Cross reference in `intro.md`.

Figure 2: Example of cross reference.

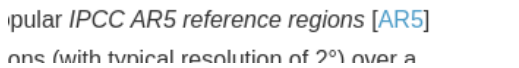
To refer bibliographic cite, `{cite}‘label-of-cite‘` has to be used. If only the author is wanted to be seen, then it is needed to use:

- `{cite:authorspar}‘label-of-cite‘`

Figure 3 illustrates how it is been used in the code (a), rendered text (b) and as a entry in the bibliography index (c).



(a) How to write a cite.



(b) How it is shown in the text.

(c) How it is shown in the bibliography.

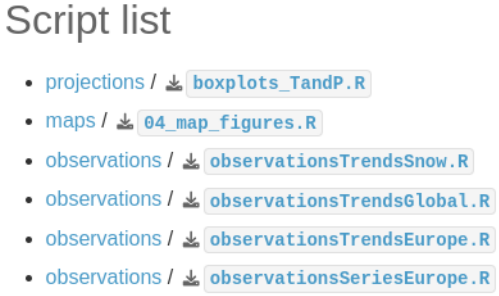
Figure 3: Example of a cite.

In some cases it is desirable to offer the possibility to download a certain file that is referenced (as a download link). To do that, the "download" role is needed. To use it, `{download}‘name-shown<path-to.the-file>‘` has to be written. An example of this type of reference can be seen in Figure 4

```
## Script list

* [projections](projections) / {download}`boxplots_TandP.R<../reproducibility/projections/boxplots_TandP.R>`
* [maps](maps) / {download}`04_map_figures.R<../datasets-interactive-atlas/04_map_figures.R>`
* [observations](observations) / {download}`observationsTrendsSnow.R<../reproducibility/observations/observationsTrendsSnow.R>`
* [observations](observations) / {download}`observationsTrendsGlobal.R<../reproducibility/observations/observationsTrendsGlobal.R>`
* [observations](observations) / {download}`observationsTrendsEurope.R<../reproducibility/observations/observationsTrendsEurope.R>`
* [observations](observations) / {download}`observationsSeriesEurope.R<../reproducibility/observations/observationsSeriesEurope.R>`
```

(a) How the "download" role is written.



(b) "Download role" when rendered.

Figure 4: Example of how the "download" role works.

By default, JB puts every bibliographic reference in the main page of the Project. To avoid this and make the bibliographic index in every page, with only the references made there, the code below has to be used in every file, placing it where it is wanted in the text:

```
```{bibliography}
    :filter: docname in docnames
```
```

2.4 Building the Project

In order to build the Project several things have to be consider: First, the building environment has to have every software needed for the JB ecosystem to function, and second, every software that is used in the code written inside the `.ipynb`files (if there is code) has to be in the environment. One of the best, and the current way to do this, is to create a *Conda* environment.

The first step to build the Project is to install *Conda*. There are several implementations of *Conda* to chose from: *Miniconda*, *Anaconda*, *Miniforge*, *Mambaforge*,... (*Conda-Forge/Miniforge: A Conda-Forge Distribution.* 2023) *Mambaforge* was used because it can download several files at once and because it uses the *Conda-Forge* repositories, in which every software version is conserved, bringing stability to the Project. To install it the right `.sh` executable has to be downloaded, then executable rights have to be enable using `chmod +x "path to the file"` in the terminal, and finally, executed using `./"path to the file"`.

Once *Conda* is installed, the next step is to create the environment for the Project. Inside the files of the Project there is a file called `environment.yml`, which has every software needed. To pass the file to *Conda* for the creation of the environment and to activate it, the code below has to be used:

```
conda env create -n atlas -f binder/conda/environment.yml
conda activate atlas
```

The first one creates an environment called "atlas" (it can be any other name) passing the location of the `environment.yml` to *Conda*, and the next one activate that environment.

After activating the environment, it is recommended to change the working directory to the one of the Project in the terminal, in order to build it. To build it the next command has to be used:

```
jb build . --all
```

The command tells JB to build the Project located in that specific folder (hence the ".") and to build everything contained in the table of contents. JB will build everything, running the code inside the `.ipynb` files (if any) and will show a log after the build showing everything incorrect like: a code that could not run, a bibliographic references that appears in several files or just that the hierarchy of *Markdown* has not been followed carefully and the file starts with a second level title (`##`) instead of a first level (`#`).

The building process will create a folder called `_build`, with the Project being rendered as a HTML interactive book and as a `.tex` with a `.pdf`. It will create cache files in a directory called `jb-cache` with the outputs of the code cells inside the `.ipynb`files, to make next buildings of the Project faster.

2.5 Repository for a Control Version System

In order to host the Project and to allow anyone to see and interact with the code inside, a repository in *GitHub* was created. *GitHub* is a platform that hosts repositories using *Git* technology. This enables the user to have a control version system for his development. *GitHub* also has some added features that enhance the process of receiving feedback and for automating the building process. *GitHub* enables the user to synchronize remote repositories, use *GitHub Actions* (explained later), and allows the community to create issues for proposed improvements and bugs, and to make pull requests with code updates, among others.

Each *GitHub* repository can have different branches. A branch in *GitHub* is a "container" that allows the user to separate different developments, i.e. the beta version of a project, that do not interfere with other branches, making easy to develop a project without making changes to other versions. For this Project, there are two branches: the main one, called "jb-test", in which the files are stored; and the "gh-pages" branch, which has the web deployment of the Project.

GitHub Pages is a feature of *GitHub* that allows the implementation of static content using HTML, CSS and JavaScript taken from a *GitHub* repository and can build and publish the result. (*Acerca de GitHub Pages - Documentación de GitHub* 2023)

Inside this repository there are several features implemented that ease the process of publishing the web version and to receive feedback from every user.

GitHub Actions

GitHub Actions is a tool to automatize any process of a project. It can, for example, compile the code written, generating an executable file and updating the master branch of the repository from another branch.

In the particular case of this Project, there are two actions implemented:

- The first one, "deploy-book", loads a generic *Ubuntu Docker* image with *Miniconda*, creates the *Conda* environment and builds the Project.
- The second one, called "pages-build-deployment", automatically updates the "gh-pages" branch once the "deploy-book" process is finished.

This actions are automated so when an update on the repository is pushed it triggers the processes of building and publishing. Listing 6 shows the actual file with the configurations of the actions.

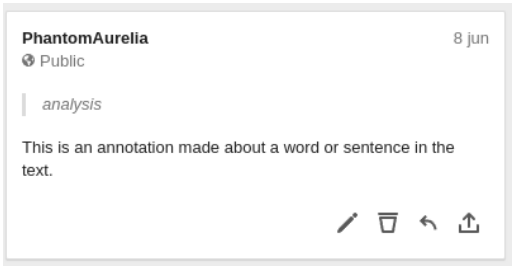
```yaml
name: deploy-book
# Only run this when the master branch changes
on:
  workflow_dispatch:
  push:
    branches:
    - jb-test
# This job installs dependencies, build the book, and pushes it to `gh-pages`
jobs:
  deploy-book:
    runs-on: ubuntu-latest
    defaults:
      run:
        shell: bash -l {0}
    steps:
    - uses: actions/checkout@v2
    # Install dependencies
    # Install Mambaforge
    - uses: conda-incubator/setup-miniconda@v2
      with:
        environment-file: binder/conda/environment.yml
        miniforge-version: latest
        miniforge-variant: Mambaforge
    # Build the book
    - name: Build the book
      run: |
        jupyter-book build .
    # Push the book's HTML to github-pages
    - name: GitHub Pages action
      uses: peaceiris/actions-gh-pages@v3.5.9
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: ./_build/html
```

Listing 6: `.yml` file with the configuration of the GitHub Actions.

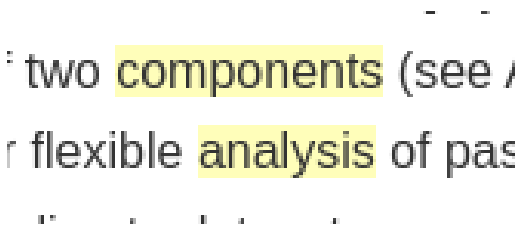
Hypothesis y Utterances

Another features implemented in the repository are *Utterances* and *Hypothesis*.

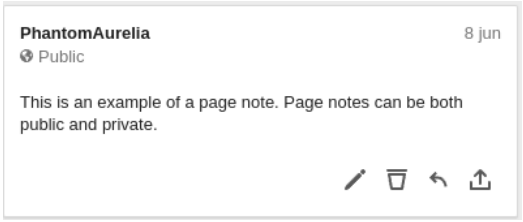
Hypothesis is an implementation in the web version of the Project that allows the creation of notes, both private and public. This feature enables the manager of the Project and any other user to interact with the rest of the users annotating further explanations. This explanations can be done using highlights, annotations in words or sentences and page notes. An account is needed to interact with *Hypothesis*. To enable this feature, `hypothesis: true` has to be written under the "html" section of the configuration file (see Listing 2).



(a) Example of an annotation.



(b) Example of highlights.



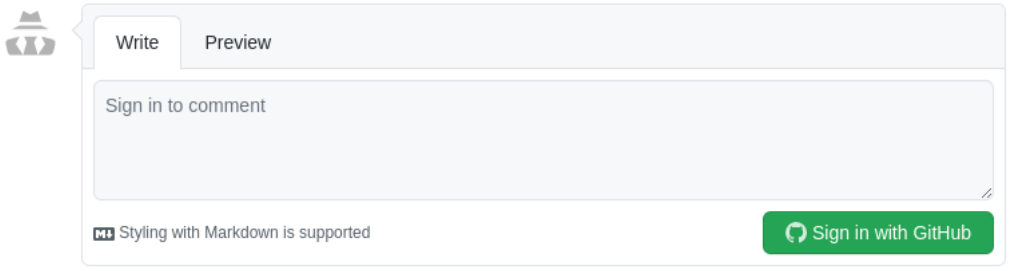
(c) Example of a page note.

Figure 5: *Hypothesis* interface in the web implementation.

Utterances is a feature that implements the ability to make issues in the Project that appear as comments in the web implementation. It is located at the bottom of every page. To use it, having a *GitHub* account is mandatory. To enable this feature, it is necessary to visit the *Utterances* web page (*Utterances* 2023) and configure it. That will generate a HTML code that can be placed in whatever file is needed. Then, it is needed to activate it in the configuration file of the Project (see Listing 2).

```
<script src="https://utteranc.es/client.js"
        repo="PhantomAurelia/Atlas"
        issue-term="pathname"
        theme="preferred-color-scheme"
        crossorigin="anonymous"
        async>
</script>
```

(a) Code written in the file for the implementation of *Utterances*.



(b) Web interface of *Utterances*.

Figure 6: Example of how *Utterances* work.

2.6 Executing the Project Interactively

The Project has a repository that enable the user to execute it remotely via the web, but can also be executed locally. Those are the two main methods of interacting with it.

To execute it locally, it is necessary to clone the repository in the local system, and follow the steps in Section 2.4.To launch it, it is only necessary to move to the Project directory in the terminal and write `jupyter lab`. This will bring a browser window with the environment *JupyterLab* ready to work with it. This method can be used to execute the code inside the Project and make changes to the files. It is more meant for the creation process, so it has to be used cautiously.

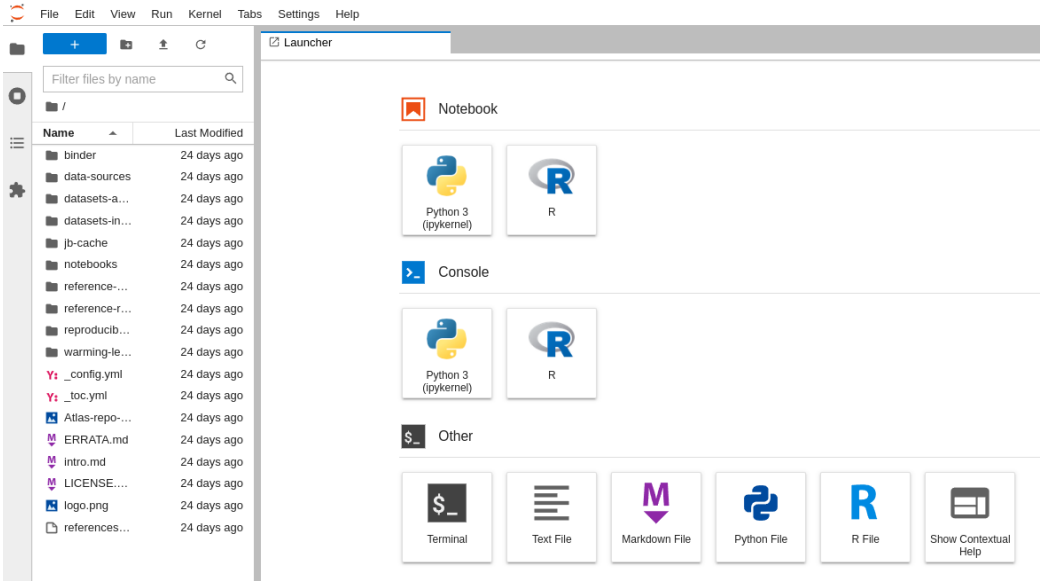


Figure 7: Jupyter Lab interface

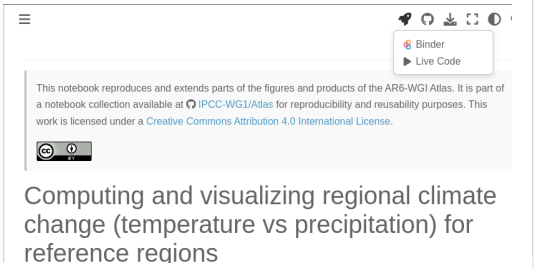
Binder and Live Code

The remote method is to use the web page of the Project and launch *Binder* or *Live code*. Binder is a platform that enables the user to create interactive environments, with the use of a *Docker* image (*The Binder Project* 2023). Once there, the code written in the `.ipynb`files can be run and modify without making changes in the original project. This is an easier way because

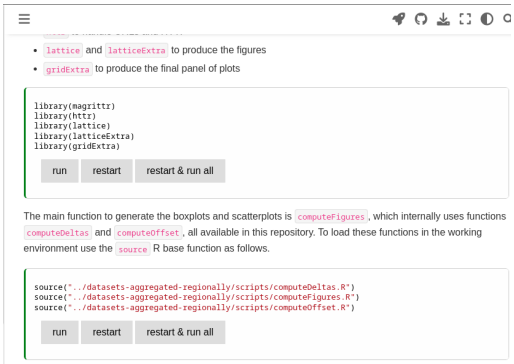
there is no need to clone a repository or build anything. It can be executed in any modern device with access to an internet connection.

With the activation of the *Thebe* option in the configuration file of the Project (see Listing 3) the *Live Code* environment can be activated. This a mix between the normal view of the web page of the Project and the *Binder* environment. It allows the user to interact with the rendered HTML and the code cells, without leaving the page.

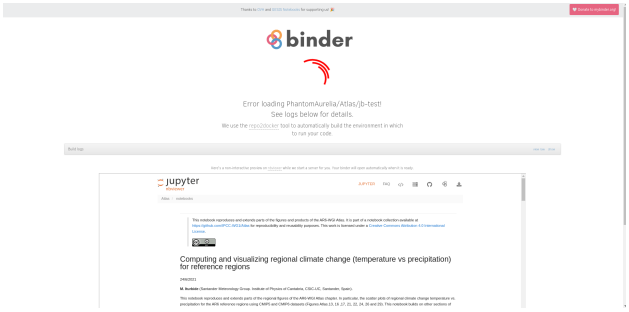
Both *Binder* and *Live Code* needs the Project to be in a *GitHub* repository in order to work properly.



(a) How *Binder* and *Live Code* can be activated.



(b) *Thebe* environment.



(c) *Binder* loading interface.

Figure 8: Example of how to start *Binder* and *Live Code* environments.

2.7 Problems Found in the Process

During the process of creating this Project, several problems were found. They have been sorted depending on whether they have been solved or not:

Solved problems:

- The original cross reference between files were made for the original *GitHub* repository (*IPCC-WG1/Atlas: Repository Supporting the Implementation of FAIR Principles in the IPCC-WGI Atlas* 2023) and everything had to be changed in order for them to function properly.
- Every type of file that was not an `.ipynb`, a `.md` or `.html` file could not be included in the JB table of contents. Those are the only type of local files that can be in the Project, and every other type of file that was considered essential had to be referenced with the "download" role.
- Every directory or file in the Project directory with a `.` at the start of the name was ignored by *Git* by default, so every change in the script for the *GitHub Actions* was not updated in the repository. The solution was to select specifically the `.github` folder to be updated.
- Due to an undetermined issue in the making process, the `.ipynb`files, which have a tag for JB to know which programming language is used the code inside, were changed randomly. The solution was to go one by one ensuring the tag was correct.

Unsolved problems:

– The most important one, and the one that has made impossible to make this report with *JupyterBook*, is the bibliography problem with *Sphinx*(*Author, Year Citations · Issue #1090 · Executablebooks/Jupyter-Book* 2023). The references in the text were correct, but the bibliography list had a weird index where every entry had a tag with a combination of capital, non-capital letters and numbers. Being this report a formal writing, that was a definitive reason to not do it in *JupyterBook*. That problem is *Sphinx* related and it has been there for several years(*Change Inline Citation Format? · Issue #201 · Mcmtroffaes/Sphinxcontrib-Bibtex* 2023). Figure 9 shows how the bibliographic index is shown in the web implementation of the Project.

Figure 9: How the bibliographic index is shown.

3 Final Result

The final result is an interactive Project which its main way of functioning is a web implementation. It can be accessed at `https://phantomaurelia.github.io/Atlas/`. The Project repository is public and can be accessed, cloned and forked at `https://github.com/PhantomAurelia/Atlas`. Figure 10 shows the main page of the project.

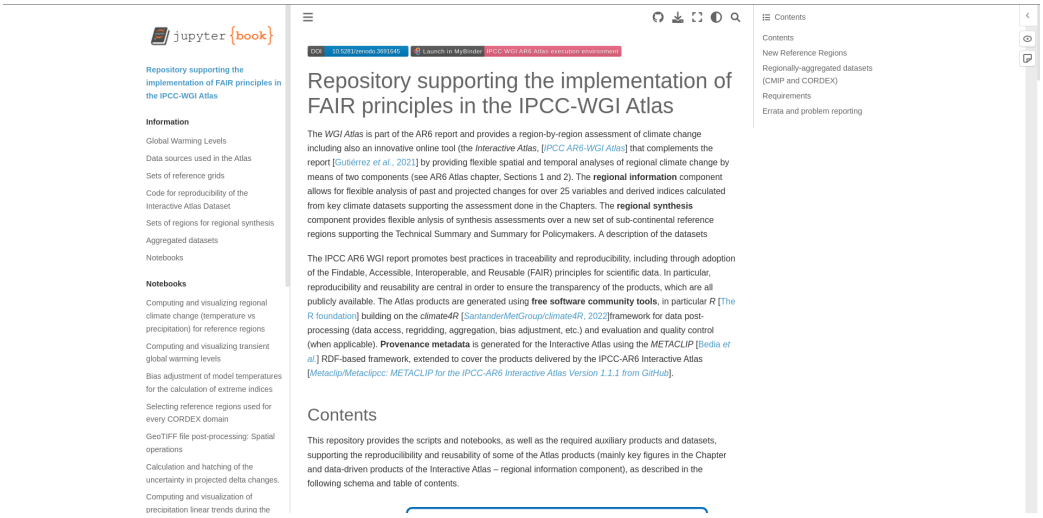


Figure 10: Main page of the Project.

Starting from left to right (using the view of the main page), first there is a menu with the table of contents of the Project (Figure 11). This menu reflects the structure of the Project, as explained in Section 2.2.

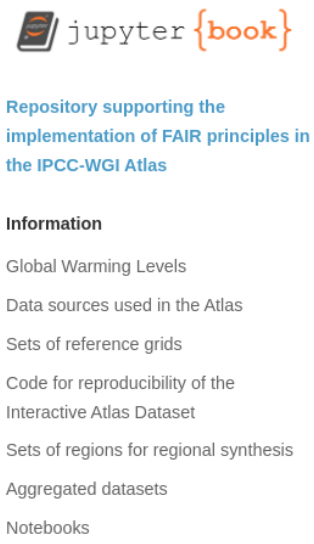
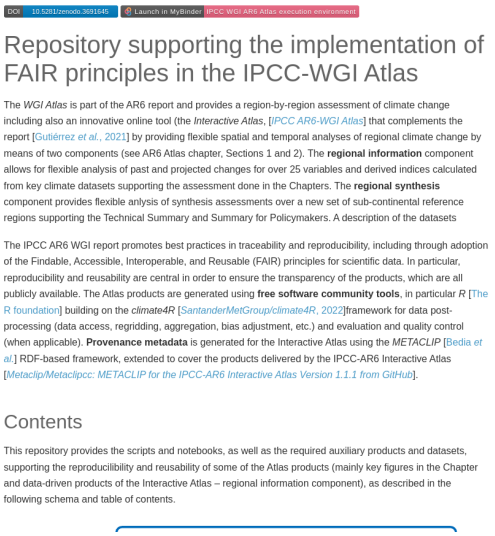


Figure 11: General Index for selecting the desired file.

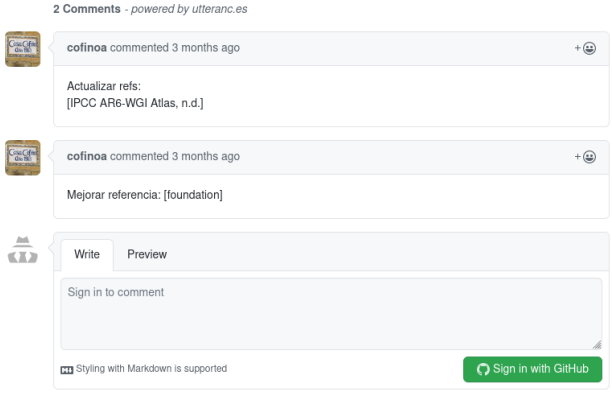
At the center of the web page, there are the contents of the different chapters (i.e. `.ipynb` and `.md` files). There is also a toolbar at the top with clickable buttons, actions related with the content of the page. Buttons from left to right in Figure 12b are as follows: a button to access the *GitHub* repository, a button for downloading the current page as a `.md` or `.pdf` file, a toggle for full-screen, a toggle for dark and light themes and a search button. If the content of the page has computational elements, an additional button will appear to launch the content of the page via *Binder* or *Live Code*. The *Utterances* interface sits at the bottom of every page as the user scrolls down. As said before, it is a simple way to open issues, add comments, etc.



(a) Main view of the chapter selected.

(b) Toolbar for chapters with non-computational content.

(c) Toolbar for chapters with computational content.



(d) *Utterances* interface at the end of the chapter.

Figure 12: Main view of the chapter, toolbar and *Utterances*.

On the right side bar there is an index of the different sections of each

chapter, as shown in Figure 13. Each section can be accessed by scrolling or clicking the name of the section in the index.

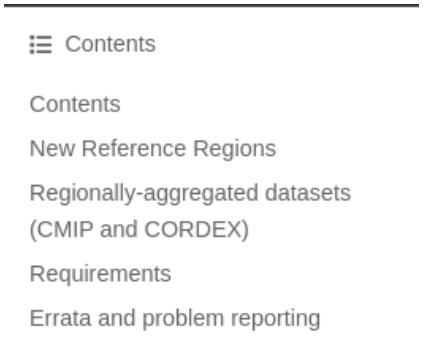
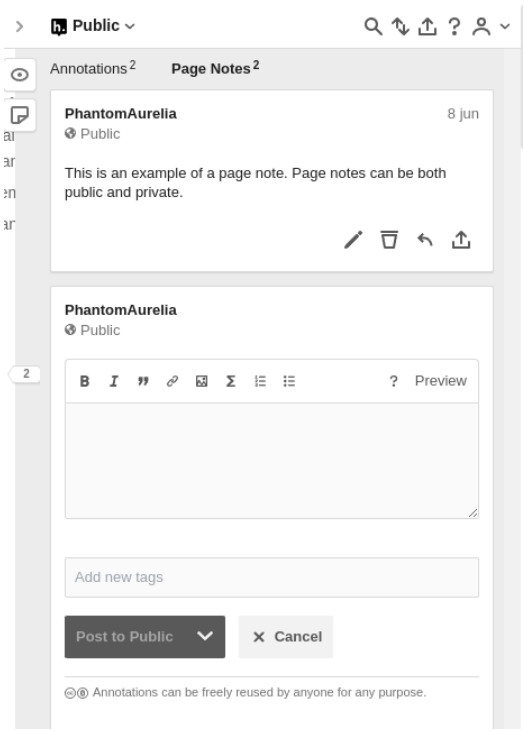


Figure 13: Index of sections in the file.

Finally, on the top right corner there is the *Hypothesis* interface (Figure 14). By default the interface is folded and by clicking the arrow button the interface expands. In Figure 14a the three buttons, from top to bottom, are: a button for expanding the interface, a button to toggle annotations on and off and a button to create a new page note.

(a) *Hypothesis* interface contracted.



(b) *Hypothesis* interface expanded.

Figure 14: *Hypothesis* interface.

Pending Tasks

The live environment as it is now does not work, due to the *Docker* image being quite heavy and old. The site in with it was placed seems to delete the *Docker* image's cache if they are not regularly updated. The process of making a new image was out of the goals of the Project, so it remains as a pending task.

Another improvement for the Project is the implementation of interactive notebook widgets, that could allow the use of graphics with zoom-in and zoom-out, different representation of plots with metadata included in every dot plotted, etc.

The last improvement would be to generate a *Docker* image to use for the automated building process in *GitHub Actions*, to make it faster. The current building process has to install *Conda* and create the Project environment each time the action triggers, which it is quite a tedious task.

4 Conclusion

JupyterBook is an ecosystem for building publication-quality books from computational material, with the possibility to implement add-ons for dynamic graphics, an interactive environment, etc. It enables the process of merge text and code in a comprehensive and cohesive way, enhancing the reproducibility and reusability. Used in combination with a repository management system like *GitHub*, it ease the control of versions of the Project and enable a continuously updated traceable publication.

Science has been democratized up to the point that in most of the countries there are several investigations made every year. The traceability and reproducibility of the investigations and reports is becoming a difficult task, and making it easier is a must.

Moreover, science has lost, in many senses, the ability to transmit information, not only inside the scientific community, but also to the rest of people. Disinformation and fake news are now a daily occurrence, and part of that is because of the lack of ability of experts in the matter to ensure that the correct information is given to the media and the rest of the people in an easy, cohesive and comprehensive way. *JupyterBook* can be used for much more than academic reports or books: it is a very powerful tool to generate interactive books for students, with virtual laboratories inside for which students can prove their skills, find out if their homework is correct and have a different learning process in their daily study. With the creation of interactive books, corrections can be done fast, avoiding students to have incorrect formulas or problem results due to forgetting a correction said by the professor in class.

Those challenges have been tried to be addressed with the edition and publication of the results of the research used for taking decisions from third parties as the *IPCC*. The *IPCC Atlas* purpose is to assess the science behind those decisions following the *FAIR* principles and the traceability and reproducibility requirements. For that, the *IPCC Atlas* was archived in a repository with every code, data and documentation used. This information was dispersed and, in some degree, disconnected. The creation of a project that contained everything in a comprehensive way was needed.

In this Final Degree Project the whole collection of code, data and documentation from the *IPCC Atlas* has been merged into a unique book binding everything as a comprehensive reference. Moreover, this Project it is not just a piece of narrative resource, but an interactive computational content which

allows the reproduction of the computed results. It also allows the reader to execute the existing code, both locally and remotely via a web environment. It enables a continuous released process integrated in a version control system as any other software project, receiving feedback from the community.

This Project has several expectations which were hard to comply.

On the one hand, one of the effects of the creation of the Project is the unification of every bibliographic and cross reference. This centralized system simplifies the management and its use to the reader. As it has been seen in Section 2.7, this has also issues that have to be solved.

On the other hand, the size of data and documentation has been a real challenge. The process of creating the computing environment was a tedious task due to the huge amount of software libraries and packages needed. These high demanding factors limit the reproducibility and traceability requirements, making them only accessible to users with big computing resources. Therefore, the process of building the Project via the *GitHub Actions* usually ends in a temporary error making the publication process more difficult. One solution to this drawback is to provide access to bigger computational resources making it more agile.

The process of getting every knowledge to create this Project have required researching and learning technologies unknown before. *JupyterBook*, *Markdown*, *Conda*, *Git*, among other technologies and skills, are used by the vast majority of the scientists on their laboratories nowadays, but they are not learnt along the undergraduate subjects. That have required the acquisition of them during the process of the creation of this Final Degree Project. The lack of knowledge of the science behind the data analysis and the different software libraries involved have been another limiting factor for a better edition and improvement of the actual contents of the Project.

Finally, this Final Degree Project has allowed the discovery of useful technologies that improve the process of doing science, not only in its diffusion but also in its development. This technologies can also improve the learning process for students, creating digital laboratories in which they can check their skills.

List of Figures

List of Source Codes

References

History of Climate Change Science - Wikipedia (2023). URL: https://en.wikipedia.org/w/index.php?title=History_of_climate_change_science&oldid=1144409600 (visited on 05/22/2023).

Manabe, Syukuro and Richard T. Wetherald (2023). "Thermal Equilibrium of the Atmosphere with a Given Distribution of Relative Humidity". In: (). DOI: 10.1175/1520-0469(1967)024<0241:TEOTAW>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/atsc/24/3/1520-0469_1967_024_0241_teotaw_2_0_co_2.xml (visited on 06/13/2023).

About — IPCC (2023). URL: https://www.ipcc.ch/about/ (visited on 05/22/2023).

Sixth Assessment Report — IPCC (2023). URL: https://www.ipcc.ch/assessment-report/ar6/ (visited on 05/22/2023).

IPCC WGI Interactive Atlas (2023). URL: https://interactive-atlas.ipcc.ch/ (visited on 06/07/2023).

The Jupyter Book Community (Feb. 12, 2020). *Jupyter Book*. Version v0.10. Zenodo. DOI: 10.5281/ZENODO.2561065. URL: https://zenodo.org/record/2561065 (visited on 05/07/2023).

Conda — Conda Documentation (2023). URL: https://docs.conda.io/en/latest/ (visited on 05/07/2023).

Zotero | Your Personal Research Assistant (2023). URL: https://www.zotero.org/ (visited on 05/07/2023).

Visual Studio Code - Code Editing. Redefined (2023). URL: https://code.visualstudio.com/ (visited on 05/07/2023).

GitHub (2023). *GitHub: Let's Build from Here*. GitHub. URL: https://github.com/ (visited on 05/07/2023).

dwhly (2023). *Home*. Hypothesis. URL: https://web.hypothes.is/ (visited on 05/07/2023).

Utterances (2023). URL: https://utteranc.es (visited on 05/07/2023).

The Binder Project (2023). URL: https://mybinder.org/ (visited on 05/07/2023).

Docker: Accelerated, Containerized Application Development (2023). URL: https://www.docker.com/ (visited on 05/07/2023).

Create Your First Book (2023). URL: https://jupyterbook.org/en/stable/start/your-first-book.html (visited on 05/07/2023).

IPCC-WG1/Atlas: Repository Supporting the Implementation of FAIR Principles in the IPCC-WGI Atlas (2023). URL: https://github.com/IPCC-WG1/Atlas (visited on 05/07/2023).

Iturbide, Maialen et al. (Aug. 9, 2021). *Repository Supporting the Implementation of FAIR Principles in the IPCC-WGI Atlas*. Zenodo. DOI: 10.5281/zenodo.5171760. URL: https://zenodo.org/record/5171760 (visited on 05/11/2023).

Plotly (2023). *Plotly: Low-Code Data App Development*. URL: https://plotly.com/ (visited on 05/07/2023).

Ven, Bryan Van de (2023). *Bokeh*. URL: https://bokeh.org/ (visited on 05/07/2023).

Jupyter Widgets — Jupyter Widgets 8.0.5 Documentation (2023). URL: https://ipywidgets.readthedocs.io/en/stable/ (visited on 05/07/2023).

Roles — Sphinx Documentation (2023). URL: https://www.sphinx-doc.org/en/master/usage/restructuredtext/roles.html (visited on 05/17/2023).

Conda-Forge/Miniforge: A Conda-Forge Distribution. (2023). URL: https://github.com/conda-forge/miniforge (visited on 05/22/2023).

Acerca de GitHub Pages - Documentación de GitHub (2023). URL: https://docs.github.com/es/pages/getting-started-with-github-pages/about-github-pages (visited on 06/07/2023).

Author, Year Citations · Issue #1090 · Executablebooks/Jupyter-Book (2023). URL: https://github.com/executablebooks/jupyter-book/issues/1090 (visited on 06/06/2023).

Change Inline Citation Format? · Issue #201 · Mcmtroffaes/Sphinxcontrib-Bibtex (2023). URL: https://github.com/mcmtroffaes/sphinxcontrib-bibtex/issues/201 (visited on 06/06/2023).

Glossary

BiBTex (`.bib`) Auxiliary software to manage the bibliography in LaTeX. 7

Conda Package and virtual environment software management system. 8, 20, 22, 33

CSS Cascading Style Sheets is what applies a style to the HTML content of a web page. 9, 14, 21

Docker Platform to build, deploy and share containerized applications. 9, 22, 25, 33

HTML HyperText Markup Language is the base of the web. 9, 14, 21, 23, 26

***Intergovernmental Panel on Climate Change* (*IPCC*)** United Nations body for assessing the science related to climate change. 6, 34

***IPCC WGI Interactive Atlas* (*Atlas*)** Tool for spatial and temporal analyses of much of the observed and projected climate change. 5, 6, 16

JavaScript Interpreted programming language for font-end. Usually used in web implementations. 9, 21

Jupyter Notebooks (`.ipynb`) An interactive computational environment that combines code, enrich text, plots, grafs, mathematics and more. 7

***JupyterBook* (JB)** A software ecosystem which is the base of the book project. 7, 8, 10, 12, 14, 19, 20, 27

JupyterLab IDE environment to create and interact with `.ipynb` files. 25

Markdown (`.md`) A markup language used to create formmated text via plain text. 7, 20

MyST Markdown A superset of the common Markdown, inspired by the RMarkdown language. 7

***Sphinx Engine* (SE)** A Python documentation generator that ease the creation of documents. 7, 18

YAML (`.yml`) A human-readable data-serialization language, commonly used in configuration files. 7

Zotero Software for managing bibliographic references.. 8