Scientific Computing: integrating the ATLAS Virtual Machine, ATLAS Open Data Jupyter Notebooks and the Git and Zenodo platforms for Open Science and Reproducibility

María J. Ramos[1] and Andrea Tugores[2]

[1] Universidad de Los Andres, Mérida, Venezuela.
author@ipvc.pt
[2] Universidad Central de Venezuela, Caracas, Venezuela.
andreatugores@gmail.com

Abstract. Taking advantage of the open science approach of the ATLAS Open Data project, which introduce students in how to perform data analysis in high-energy physics, the project was focused on the improvement of the ATLAS Open Data Jupyter Notebooks with a Python framework script: by automating, making the running of cells more interactive with the user, and by expanding the documentation make them easier to understand. And as part of encouraging Open Science with the tools that ATLAS, CERN and Git offer, an assistant program that integrates the Git commands and Zenodo REST API was designed to be used in the ATLAS Virtual Machine, for promoting the concepts of open science and reproducibility.

Keywords: ATLAS Open Data · ATLAS Virtual Machine

1 Objectives

- Improve the 13 TeV ATLAS Open Data Jupyter Notebooks by adding explanations on both, the physical analysis and the code of the Jupyter Notebooks that contain Python code.

- Create a new Jupyter Notebook that is not focused on a particular analysis, but has a more general approach that can provide the appropriate information of the specific analysis when needed.

- Make a more interactive experience in the Jupyter Notebook by requesting inputs from the user to proceed with the analysis and plotting.

- Provide more reachable documentation of ATLAS Open Data for the users of the ATLAS Open Data Jupyter Notebooks.

- Integrate the ATLAS Virtual Machine as a useful platform to run the ATLAS Open Data Jupyter Notebooks by promoting its use in the README file of the repository for the new Notebook.

- Propose a new way to teach the importance of Git and the usage of git commands in the linux terminal of the ATLAS Virtual Machine as a way of version control.

- Promote GitHub, GitLab and Zenodo as platforms for Open Science and the advantages of using these websites.

2 Introduction

ATLAS is one of the major experiments at the Large Hadron Collider (LHC) at CERN. It is run by the ATLAS Collaboration, an international team of physicists, engineers, technicians and support staff. The interactions in the ATLAS detectors create an enormous flow of data, which is then analyzed by the ATLAS Collaboration. However, it is possible to take a look at some of their data and analysis thanks to an initiative to release datasets called ATLAS Open Data, an educational project that provides data and tools to high-school, masters and undergraduate students, as well as teachers and lecturers, to help educate them in the physics analysis techniques used in experimental particle physics.

One of the main data files of ATLAS Open Data is the 13 TeV samples, which were released in 2016. As part of the resources available to the public, they are also accompanied by a set of Jupyter notebooks. A Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. The notebook extends the console-based approach to interactive computing, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results.

The ATLAS Open Data Jupyter Notebooks allow data analysis to be performed directly in a web browser by integrating the ROOT framework with the Jupyter notebook technology, a combination called "ROOTbook". The frameworks implement the protocols needed for reading the datasets, making an analysis selection, writing out histograms and plotting the results, by only needing to execute the code written in the Jupyter Notebook, taking advantage of its functionalities.

The project is encapsulated in the ATLAS Open Data project, by following and sharing the philosophy behind it. To understand its objectives, it is necessary to know some important concepts that have revolutionized the scientific world, where researchers are meant to share information in an open manner in order to foster a more scholarly environment, needing Open Science and Reproducibility.

As open access, open data, open source and other open scholarship practices are growing in popularity and necessity, they have enable analysis to be taken to the next level in order to further scrutinize the data. This caused the Open Science movement to emerge from the scientific community and spread rapidly to other disciplines, calling for the breaking down of barriers to knowledge, to promote and advance scientific research and discovery. Open science can be defined as a collaborative culture enabled by technology that empowers the open sharing of data, information and knowledge within and outside the scientific community to accelerate and make scientific research more accessible.

On the other hand, one of the pathways by which the scientific community confirms the validity of a new scientific discovery is by repeating the research that produced it. From this practice of repeating an investigation, the terms of reproducibility and replicability emerge. Reproducibility means computational reproducibility, obtaining consistent computational results using the same input data, computational steps, methods, code, and conditions of analysis. Replicability means obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data[5].

Because the traditional methods section of a scientific paper can be insufficient to convey the necessary information for others to reproduce the results, due to large volumes of data and complex code, tools such as Git and Zenodo have emerged to facilitate the pathways to open science and reproducibility. [5].

Git is an open-source version control system. Version control systems keep revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

The other tool, Zenodo, is a general-purpose open-access repository developed under the European OpenAIRE program and operated by CERN. It allows researchers to deposit scientific content of any kind assigning in the process a Digital Object Identifier (DOI), which means that it is not needed to publish in a journal in order to obtain a unique and permanent DOI, making publications more easily citable according to international standards. Other features are very helpful for researchers: flexible licensing and accessibility, long-term storage, automatic integration in reporting for European Commission-funded projects via OpenAIRE.

In a way, ATLAS Open Data, Git and Zenodo are means to achieve the same end: promote Open Science and Reproducibility. But as they can be used to focus on different parts of the process of making scientific research, including its datasets and code, as available as possible, there is a need to integrate all these tools. By connecting resources, one is almost sure that an even brighter future

is yet to come for Science and humanity.

3 Problem Statement

Although the ATLAS Open Data website, resources and tools are extensive and well-planned, as the ATLAS Open Data project grows and looks for the improvement of its contents, looking for its flaws and updating its material has become almost a requirement [1].

For the correct execution of an ATLAS Open Data Jupyter Notebook, the use of ROOT is necessary, however it was not imported in the beginning of the code for the Python framework-scripts, causing errors in the cells that ordered the run of the analysis. ROOT is a modular scientific software toolkit and provides classes for big data processing, statistical analysis, visualization and storage. As this tool is required, the ATLAS project provides a virtual machine that integrates ROOT into Jupyter Notebook.

The last update of the ATLAS Open Data Jupyter Notebooks, although containing the necessary code to run any of the 12 analysis from the 13 TeV ATLAS Open Data physics analysis examples, was focused on presenting two analysis of the data corresponding to the HZZAnalysis and the HyyAnalysis folders. To access the other examples, the user had to modify the code, and although it represented insignificant changes, it could always end up in errors and unexpected outputs.

The HZZAnalysis notebook initially showed a histogram that was uploaded in a markdown cell, this histogram corresponded to only one of the many generated when performing the HZZAnalysis analysis. Performing any of the other analyzes would show the same histogram and that was also the case of the HyyAnalysis Notebook, containing and showing just a Hyy plot.

In addition to that, there was a lack of the proper documentation inside the notebooks, both from the physical aspects of the analysis and from the commands in the code. These weak points meant a longer process of learning for the less experienced and knowledgeable users of these Jupyter Notebooks series.

On a side part, even when ATLAS has a platform to create code, run analysis, and get data from their ATLAS Open Data, which is the ATLAS Virtual Machine (VM), this was not fully integrated to the Jupyter Notebooks in the sense that the mention of the ATLAS VM did not describe it in an accurate way, that presents it as an extraordinary tool to run the ATLAS Open Data Jupyter Notebooks and to later created original content from ATLAS Open Data. The ATLAS VM was also not introduced clearly in the README file. For students who accessed the ATLAS GitHub repository before the ATLAS Open Data web-

site, the information and tutorials of the VM would remain unknown until they navigate to the other sites.

Finally, even though the philosophy behind ATLAS Open Data is about Open Science and Reproducibility, there is no information available for the user of at least some platforms to make science more shareable and discoverable. Even when Git is a well known software for version control that facilitates scientific reproducibility, not all students are Git users, and not all students are familiar with executing commands in a terminal, or have a complete understanding of the basic git commands. The ATLAS Open Data Jupyter Notebooks are great educational tools but they work best for users who are already GitHub users, and have already had experience with the functioning of the git commands in their local machine.

Going beyond the code, the need of sharing, citing and the DOI versioning in the scientific community is constantly growing. Although it is not required, or needed, for the ATLAS Open Data website to include options for scientists to help in the publishing of their contributions, CERN is part of the organizations that fund Zenodo. It would be coherent for the ATLAS Open Data to introduce students in training to this kind of tools to promote Open Science, as it is one of the the key concepts of the ATLAS Open Data educational project.

By considering this last point, one can understand that the many objectives of this project converge to the fact that, even when ATLAS Open Data is a vast resource of knowledge, it needs constant updating to provide the best tools for learning and emphasize the importance of Open Science and Reproducibility.

4 Methodology

Starting first with the ATLAS Open Data Jupyter Notebooks, the material to improve was the collection of Jupyter Notebooks that run the analysis frameworks at 8 TeV and 13 TeV from the ATLAS Outreach Data tools. As this collection is extensive, to narrow the content to ameliorate, the selected notebooks were the ATLAS Open Data Python 13TeV framework-script, which show two examples of physics analysis (HZZAnalysis and HyyAnalysis), executed by a similar code.

Considering interactive content as a powerful resource that provides a unique experience to users and manages to retain their attention more efficiently, an approach of making the code more interactive was decided to be the best way to improve the Jupyter Notebooks.

Python packages are the fundamental unit of shareable code in Python, and they make it easy to reuse your code; maintain it and share it with your colleagues

and the wider Python community. By taking this into account, being sure that all the needed packages were imported at the beginning of the notebook was an important step to guarantee that the analysis and plotting would take place when their corresponding commands were executed by the user.

Apart from the previous required packages, new libraries were added, such as pandas and matplotlib, to be able to use DataFrames to store information of the analysis, and to show the histograms that are produced within the same notebook.

To run the code as smoothly as possible, conditionals were included in certain cells, to avoid the repeated execution of commands that were only needed to be used only once.

And although that it can seem that all of these changes make the experience less involving to the user than what was originally planned to be, the user keeps interacting with the Jupyter Notebooks but in the form of their inputs, and not rewriting the code, which can result in errors and unexpected outputs at the end. This focuses the attention on understanding the commands and what they do.

Figures 1 and 2 show examples of interactivity with the user. In Figure 1, *while loops* are used as a tool to ensure that the user correctly enters the name of any of the possible options. The *try-except* statement can handle exceptions. Exceptions may happen when you run a program. Exceptions are errors that happen during execution of the program [4]. If the name is misspelled or numbers are entered, a message is displayed to the user if the desired file is not found. The question repeats itself until a name is entered correctly and the while loop is exited.

```
while True:
    analysis = input("Enter your choice: ")
    try:
        os.chdir("Analysis/"+analysis)
        break
    except FileNotFoundError:
        print("Analysis not found")
```

Fig. 1. Interactivity with the user to enter the name of the analysis.

In Figure 2 the user enters a number to select which analysis he wants to carry out the histograms. A few cells of code before it is explained in a markdown cell

which number corresponds to each analysis so that the user knows which number to enter to obtain the desired result. As in the previous figure, a while loop and the try-except are used to ensure that a number between 0 and 9 is entered, which are the possible options. If a string is entered, a message is printed telling the user to try again and if a number is entered that is not among the options, the user is prompted to enter a number in the range from 0 to 9.

```
while True:
    number = input('Choose your analysis. Enter a number: ')
    try:
        number = int(number)
        if 0<= number <=9:
            break
        else:
            print("Valid range, please: 0-9.")
    except ValueError:
        print("Try it again.")
```

Fig. 2. Interactivity with the user to enter the number of the analysis.

By adding interactivity with the user and storing their responses in variables that were later used in other code cells, the Notebooks were generalized to just one Notebook, without needing to have more than one example notebook to understand their functioning. The histogram problem was solved by making the display of the plot not in a markdown cell, but in the output of a code that the user can run after the histograms are created. Not only this guarantees that a plot of the chosen analysis is the one that is shown, but that all of the plots generated from this analysis are shown, Figure 3. An example of how the produced histograms would look can be seen in the Figure 4.

This work was complemented by including detailed information in the markdown cells about: the imported Python packages, their uses and functionalities; explanations of what was happening in each of the code cells; the commands and the methods function, and the physical analysis details. The newest changes allow the user to see as an output the full name of the analysis chosen; a review of the physics analysis taken from official documentation of ATLAS, and a short explanation of the plots generated. This is particularly useful to students unfamiliar with High Energy Physics.

Continuing with the approach to integrate the ATLAS Virtual Machine, the solution to this problem was simple, but not less effective: by writing a tutorial in the README file, and including the links to the ATLAS VM documentation and the already made video tutorials, the information is more reachable and

```
files = os.listdir(f"{directory4}/histograms")
for i in files:
    path = f"{directory4}/histograms/{i}"
    image = plt.imread(path)
    plt.figure(figsize=(50,30))
    plt.imshow(image)
    plt.axis("off")
    plt.show()
```

Fig. 3. After entering the number of the analysis to be plotted, the histograms are displayed.

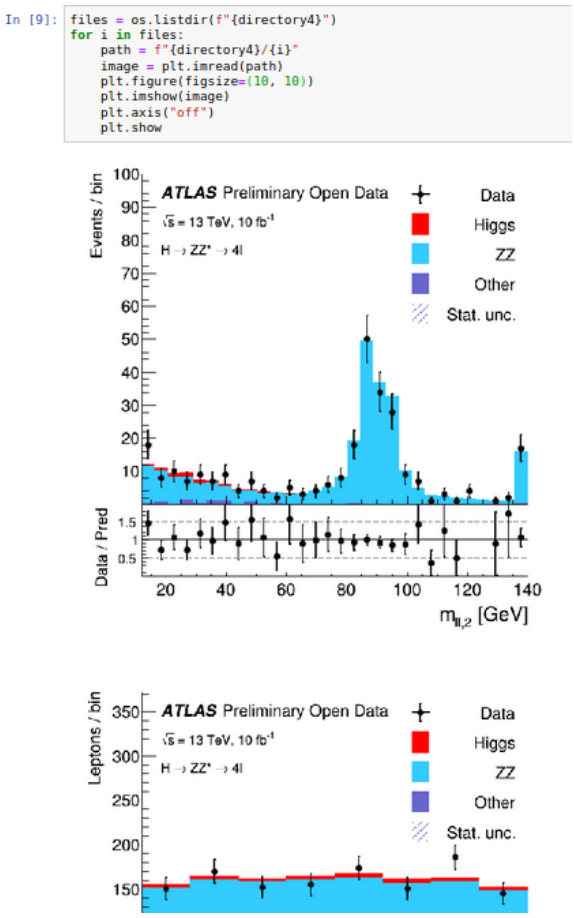

Fig. 4. Histograms.

easier to navigate in.

Ultimately, to promote the usage of Git and Zenodo, a program was created, named the Git&Zenodo Assistant. Its main menu is stored in a tutorial.py file that calls the functions in the packages and subpackages which contain the necessary code to run all the program's options.

The code for the program is written in Python3 and it requires the installation of these packages and their updates: pip, to install and manage the other packages; requests, for the codes of Zenodo Rest API; termcolor, for coloring the words that appear in the program; pandas, to manage the database of the controlled vocabulary for the metadata of Zenodo; and tkinter, (and tkcalendar) for using the graphical user interface. All of these packages are listed in a .sh file with the explicit name of requirements.sh. The instructions of installation are explained in the README file.

The program was designed to run specifically in ATLAS Virtual Machine terminal for better integration, although creating a program compatible in local machines and different operative systems could be desirable.

The true functionalities of the Git&Zenodo Assistant Program come from its packages and subpackages. They contain functions that allow the user to get links for the Git and Zenodo documentation; to review some important concepts of Git, its commands and Zenodo's features, in windows that are optional to open; to upload a file to Zenodo, including its metadata by filling some required information, and to push a file to Git not by typing git commands, but by pressing enter to execute. In case of needing help, the README file contains the links and steps to create a GitHub and Zenodo account and to generate the tokens to access the options of the program, which are necessary to git push and upload a file (Figure 6).

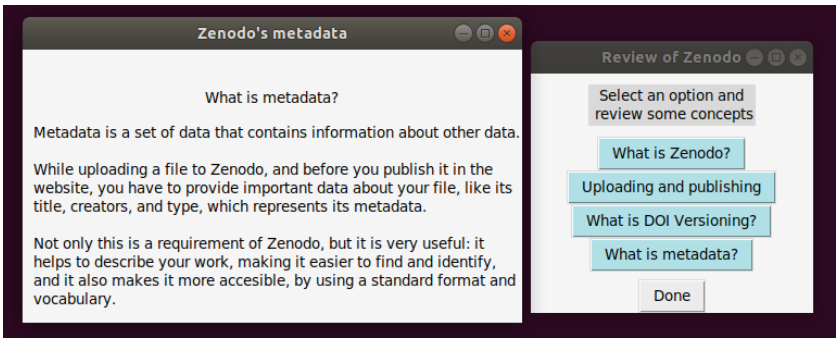

Fig. 5. Zenodo review in the GitZenodo Assistant program.

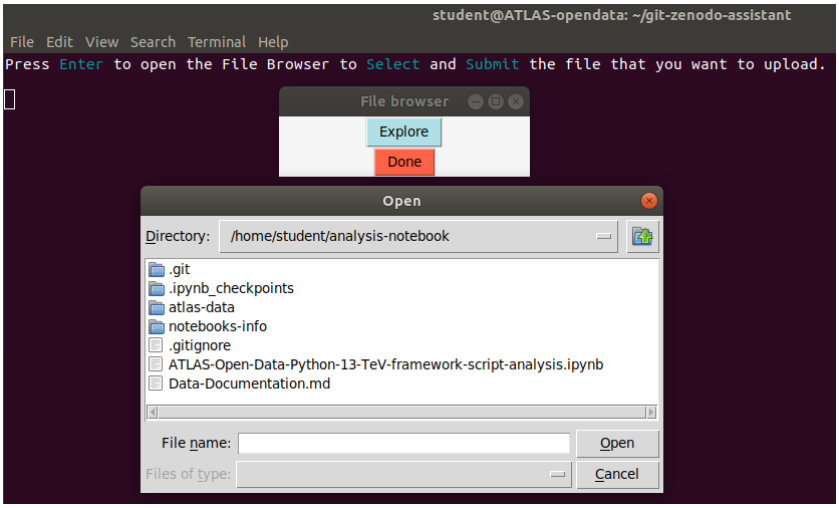

Fig. 6. Choosing a file to upload using a File Browser in the GitZenodo Assistant program, which uses tkinter.

Note that, because interactivity was considered an important part of the experience, the tkinter module was used to generate windows for: selecting a date from a calendar; choosing a directory or a file from a file browser to get a path; choosing from a list of options. All of this, while conserving the terminal experience, as the git commands appear in green letters to emphasize the importance of executing git commands in the terminal (Figure 7).

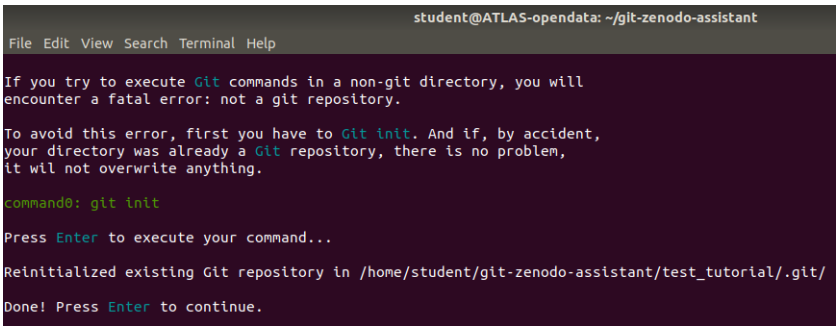

Fig. 7. Executing git commands by pressing Enter in the GitZenodo Assistant.

The repository in which the README, the Notebook and the program are stored also contains image files and markdown documents with the information that is provided in the notebook and the program, as a way to make it easier for the

user to review the information without running any code, making documentation always available.

5 Results

The notebook allows the user to have a clearer idea of what is happening as each cell is executed.User interactivity is no longer through direct modifications to the code but through inputs where the user is asked to enter any of the available options. By adding information on the physics and on the generated histograms, the user has the opportunity to access the entire analysis process without leaving the notebook, being able to even see the results.

The tutorial.py program works correctly in the ATLAS Virtual Machine terminal. When the program is executed, it presents these options to the user:

- Type “a” to enter the Git Tutorial, which does some git commands including a final git push to the user’s GitHub or GitLab account. If executed successfully, the user will not have to type any commands, but to press enter to execute them, as it was planned from the beginning.

- Type “b” to enter the Zenodo Assistant, which can upload a file to Zenodo. This operation calls the functions in the corresponding subpackages. If executed successfully, the user will not have to type any commands, but to provide the location of its file, and some basic information about the upload, to generate the metadata. The uploading is done by working with the Zenodo Rest API, although the user will not be involved directly with the codes and requests, they will only need to enter their access token

- Type “c” or “d” to see the links to the official documentation of Git and Zenodo respectively.

- Type “e” to exit the program.

All these options did not show any unexpected-outputs in the final tests. However, when executing the Git Tutorial or Zenodo Assistant, some errors can be encountered which are explained below.

While executing the Git Tutorial:

- “git remote rm origin” command could cause a fatal error in the case that there was not an origin. The program will continue to execute without problems in any case, this command was included to assure that there was no previous origin that could later cause errors.

- "git remote add origin" and "git add" could get an error as an output if the user provides the wrong url or the wrong file name. The program will not handle these errors, but to avoid them the user will be asked to check the url and file name before executing the commands.

- "git push" could get an error if the authentication data provided by the user, that is, their username, password or token are invalid. The program will not handle the error, but will provide the user another chance to git push if the operation was not successful.

The errors found while uploading a file using the Zenodo assistant are caused by requesting issues due to: invalid Token, and a Zenodo's server error. The program includes code to handle an invalid Token, asking the user to reenter their Token. In case of a server error, the program will not try to upload a file again, and will ask the user to try later. As these errors are not caused by the code of Zenodo's assistant, if they do not happen while running the program, the file will be uploaded successfully.

To reduce the probability of encountering errors, the user has to follow the instructions correctly and provide the correct data and authentication of their GitHub, GitLab or Zenodo account respectively. Also the user should also have correctly installed the requirements in the requirements.sh file before running the program. These previous steps are specified in the README file. When following the instructions and taking care before and during the running, the tests did not cause errors and managed a successful upload of files to Zenodo and GitHub.

Finally, although the running of the Git&Zenodo Assistant program in a terminal different from the one in the ATLAS Virtual Machine is not a part of the objectives of this project, to see its functionality, it was also tested in a local linux terminal and the code could also run as expected. Instead, when the code is executed in Windows, it returns some wrong outcomes. It can also be executed from JupyterLab in a linux-like terminal and it could present problems in the terminal of JupyterHub.

6 Conclusion

Considering that the work carried out focuses on promoting, for educational purposes, examples of data analysis carried out in high energy physics and seeking to make the information as accessible and easy to understand for the user, it can be said that the objectives were achieved.

First, the project was presented with extensive documentation. In the repository, the user will find: the citations from the datasets used for the analysis executed

in the ATLAS Open Data Jupyter Notebooks; links to the ATLAS Open data website, the ATLAS GitHub account, and the repositories containing the referenced analysis; links to the GitLab, GitHub and Zenodo websites, their official documentation and the citation and links to the ATLAS Virtual Machine, its tutorials, and to the VirtualBox.

Regarding the Notebook created, each of its code cells has an explanation that allows the user to understand what will happen when the cell executes, allowing those who do not have programming knowledge to have an idea of what is happening in each step. It was also improved as planned: by getting the general approach to the analysis, the user is provided with the corresponding information and plots of their chosen analysis.

In addition, with the Git&Zenodo Assistant program, the user receives an introduction to the Git and Zenodo tools, an introduction focused on the learning of their functionalities and promoting their use as important steps to the good scientific practices training of the user.

The ATLAS Virtual Machine was integrated with the new Jupyter Notebook and the Git&Zenodo Assistant program, as they were made specifically to be runned in the ATLAS Virtual Machine and the user is instructed on its use.

The development of the resources of this project opens an interesting discussion about learning, Open Science and Reproducibility: the easier it is to access information and as more tools that facilitate the sharing of new knowledge and results continue to be developed, the more interconnected the world of science will be.
When researchers are unable to access information, learning is obstructed, innovation slows and scientific progress is hampered from reaching its full potential. By emphasizing the importance of open science and reproducibility, this situation can be avoided.

Publishing in open access journals is associated with more citations causing other authors to cite more papers because they do not have to pay to read them. Sharing articles on social media and mainstream media outlets helps researchers get noticed, and this can only be done with open publications. Contrary to what some might believe, the peer review process for open access publications is quite rigorous and is now becoming transparent such that the review is published along with the paper, enabling the reader to really see what went into getting the paper published.

By submitting data and research materials to independent repositories, the content of your research is preserved and accessible for the future. This is particularly beneficial when researchers have to respond to requests for data or materials. In addition, when researchers release their data, software, and ma-

terials there is clear documentation of the key products of the research, thus increasing the reproducibility of the findings by other researchers. Ultimately, when researchers share data and materials, they value transparency and have confidence in their own research.

Recent evidence that the open sharing of articles, code, and data is beneficial for researchers has been demonstrated and is shown to be very strong. Every year more studies evaluate the open citation advantages and more funding agencies are announcing policies encouraging, requiring or specifically financing open research. In addition, more tools are available to make the sharing process easier, quicker, and more cost-effective.

To help ensure the reproducibility of computational results, researchers should convey clear, specific, and complete information about any computational methods and data products that support their published results in order to enable other researchers to repeat the analysis.

Even though the achievement of the objectives was declared at the beginning, a list of suggestions is provided at the end as proposals to continue this work. Sharing science is a job that never ends.

7 Limits and Suggestions

- Expand the explanations of the physical findings in the ATLAS Open Data Jupyter Notebooks, by creating more original content to add or replace the current one.

- Expand the explanations of the commands.

- Delve into the documentation and functionalities of ROOT to explain how histograms are produced.

- Create a FAQ segment in the ATLAS Virtual Machine tutorial section, to propose solutions for possible errors while downloading, installing and executing the ATLAS Virtual Machine.

- Improve the Git&Zenodo Assistant by making it more compatible with Windows, so it can be used outside the ATLAS Virtual Machine.

- Improve the Git&Zenodo Assistant program by adding more git commands and its functioning.

- Correct the possible errors that could be presented when executing the commands from the Git&Zenodo Assistant program.

– Generate a virtual display to use the Git&Zenodo assistant program in the JupyterHub platform.

References

1. Atlas open data, https://atlas-opendata.web.cern.ch/atlas-opendata/
2. Git, https://git-scm.com/
3. Open science for open societies, https://www.openscience.eu/zenodo/
4. Python tutorial, https://pythonbasics.org/try-except/
5. National Academies of Sciences, E., Medicine: Reproducibility and Replicability in Science. The National Academies Press, Washington, DC (2019)