H-SAF DATA EASY ACCESS AND HANDLING TOOLS

TASK II REPORT

DEVELOPMENT OF COMMON GUIDELINES FOR SCRIPT AND TOOLS DEVELOPMENT

Felix Enyimah Toffah

November 2023

Supervisors

- Dr Luca Ciabatta
- Dr Luca Brocca
- Dr Simone Gabellani
- Dr Nicoletta Roberto

Version history

This is the first version of the guideline document.

Executive Summary

This document contains guideline information for the development of Jupyter notebooks to support training on the use of H-SAF products. In addition, there is a template that could be used when developing the training materials (notebooks).

Table of contents

Chapter One

Introduction

1.1 Introduction

The EUMETSAT Satellite Application Facility on Support to Operational Hydrology and Water Management (H SAF) generates, distributes and archives several satellite-derived soil moisture, rainfall and snow products to support operational hydrological applications. These data are essential for numerical weather prediction, natural hazard monitoring and mitigation, water management and agricultural applications.

With the start of the fourth Continuous Development and Operational Phase (CDOP 4), the H SAF product portfolio has reached a high level of maturity, allowing to foster the consortium effort in the user promotion and data access activities. Considering the already wide availability of products and taking into account the exploitation of the new MTG and EPS-SG programs foreseen in the next few years, the H SAF data offer will become even richer. This poses the need to guide and help the user in the access and use of the provided products. A tool for training the user in this regard is represented by computational notebooks, which are interactive web applications that combine text, code and computational output (Wagemann et. al., 2022).

To guide users in their products, the H SAF consortium has been developing tools including computational notebooks for the training of the end-users under the clusters of snow, soil moisture and precipitation. However, there is a need to streamline the development of these computational notebooks for organisational purposes. In this contest, the main objective of this document is to guide the development of these notebooks to ensure their consistent development under the H SAF umbrella. The ultimate aim is that the content of the computational notebooks will be very educative, user friendly and instructive to the end-users.

This guide document is supposed to be a living document for the development of the user promotion tools. New best practices on programming and software development in general may be added to these guidelines in order to keep them updated.

1.2 Objectives:

Among others, the following are the main classified objectives for this document.

1. Define a set of scientific standards or guidelines that could be followed when creating Jupyter notebook files to support H SAF educational training activities. The standards have to take into consideration:
    a. all the necessary sections within a script used for H SAF purposes (i.e. data access and authentication, data download, data reading and pre-processing, data analysis and results visualisation);
    b. a detailed step-by-step description of the notebook with additional information related to H SAF, the product under consideration and the script that could be of benefit to the end-user in understanding the provided material and the use of the particular product/application;
2. Define a structural design that should be followed when creating a Jupyter notebook file, in order to improve readability and ease of usage;
3. Create sample Jupyter notebook files for the various clusters following the defined guidelines and using the materials received in Task I.

It is hoped that by following these guidelines, one would be able to create a computational notebook with the recommended contents to achieve the target objectives.

1.3 Target Audience

This guideline document is intended for all the stakeholders and the consortium partners of H SAF who will be impacted directly or indirectly by the to-be-developed notebooks under the various clusters. Mainly researchers, end-users and all other individuals that use the H SAF products and that have varying levels of knowledge and expertise in programming.

Chapter Two

Proposed Structure of the Notebooks

The following are the proposed essential elements that have to be present in any notebook file that will be developed. A template notebook that has these elements will be provided in addition to this document. Due to different nature of the H SAF products, some sections may vary in the content.

2.1 Header/ Title of the Notebook

This refers to the central theme of the notebook. It should be centred, made bold and of a font size larger than the rest of the text to make it stand out.

2.2 Breadcrumb Navigation

It is intended that the notebooks will be developed under a parent folder, containing the material from each cluster. The use of breadcrumbs as navigation aid will help the user be aware of where he or she is in the parent folder. Under the respective clusters, it is proposed for there to be a grouping of files in the classes practical and exercise: the practical folder to contain practical demonstration of usage of the data and the exercise folder to contain samples for the user to practice.

2.3 Prerequisites and Data Sources

In this section is to be stated the list of data being used for carrying out the task of the notebook. Since the data will be obtained from the FTP server, it will not be necessary to reference the data source for those obtained from the FTP server. However, for external data sources, it will be necessary to reference their sources in this section.

First, indicate the data being obtained from the FTP server and then the ones being obtained from external sources.

2.4 Table of Contents

At this section is placed the Table of Contents meant to allow easy navigation to various sections of the notebook. See Section 4.2.15 and 4.2.23 for guide on using links for creating the table of contents.

2.5 Introduction and Objectives

A brief introduction on the theme of the notebook and its scope is expected for the understanding of the user and to let him appreciate the purpose of the theme. The introduction should have a general statement about the topic in simple language, specific key points on the topic which you are going to discuss in the rest of the document and an overview of the expectation to be achieved at the end of the notebook.

Also, clearly define the objectives you set to achieve at the start of the document. This will set the stage for you to streamline the rest of the contents of the notebook. Having the objectives set at this stage will also raise the expectation of the end-users to follow the rest of the document and help them understand the purpose of the notebook.

2.6 Library Imports

A library is a group of related modules that can be used in different sections of a software program for specific operation.

For educational purposes, Wagemann et al. (2022), stress the need for imports to be brought at the beginning of a notebook. Thus, the libraries being used for the codes in the notebook are to be placed here in the order of:

- Standard library: these are built-in modules that provide access to basic system functionality like system input/output operations. Some of the python standard libraries are time, sys, os and math. For a list of other standard libraries, please follow the link here.
- Related third party imports: these are libraries developed by third parties that are not standard libraries and provide additional functionalities to a

program beyond the standard libraries. Some of these libraries are numpy, pandas, scikit-learn, matplotlib, tensorflow, django, flask, requests, etc. They are usually located in a dedicated site-packages directory of the python installation folder.

- Local application of library specific imports: these are the modules or functions that you have developed that are specific for the program you would be developing.

To improve readability of these groupings of the libraries, the import groups should be separated by a blank line.

2.7 Data Access and Authentication

In this instance, the user credentials needed for accessing the FTP server will be required. Thus, the user has to be pre-informed on the need to provide their own credentials (or ad-hoc login info created for training purposes) in order for the required data to be downloaded from the server. In this regard, the ipywidget's Password and Text widgets could be used to collect the password and username respectively. It is strongly recommended to use format strings whenever the user's credentials are being used in the rest of the program. Check to ensure that the credentials provided by the user are correct and provide appropriate error messages when the wrong credentials are provided. Also, the DatePicker widget could be used to let the user select the desired start and end dates for which to perform the studies with the notebook.

2.8 Data Download

In this section, the program to download the required data from the FTP server is to be provided. This is to be done using a combination of code and markdown cells.

2.9 Data Reading and Processing

This is the section dedicated to extraction of the data into a readable format, the reading of the data and processing of it according to the theme of the notebook.

2.10 Data Analysis/Evaluation and Plotting

In this section are to be located the analysis needed to be performed on the processed data and subsequent plotting of it. In this section will be added all the necessary analysis to be carried out for specific applications in addition to the data extraction (anomalies, modelling). Choose appropriate charts and plots for different types of data and include titles, labels and legends to enhance interpretability.

2.11 Results Visualisation

Here, the results of the plot are to be displayed.

2.12 Save the Result for Future Purposes

Future use of the results of the data processing by the end-user is anticipated. This section is to allow the user to save the results in '.png' image format and the data in '.txt', '.csv' or the 'netCDF4' format. Additional output format could be defined if required.

2.13 Conclusion

A short summary of the whole program will be very useful to conclude the notebook. Summarise your work by starting from your problem statement, moving to the approach you followed and give a summary of the results you obtained.

2.14 Next, Previous and Jump-To-Top Navigation Links

These links will be useful to enhance navigation on the page and other related notebooks.

2.15 References

Finally, reference documents used for the exercise as well as references to other useful documents are to be provided in this section for the benefit of the user.

Figure 1 gives a snapshot of the expected layout of the contents of the notebooks. The contents of these sections are to be provided using a combination of code and markdown cells. Some markdown elements have been provided in Chapter 4 of this document and could be used to improve the usability of the notebooks. The template notebook related to this document has these features developed in it. The content of the template could be modified to suit the theme under which it is being developed. However, it is recommended that the sections proposed above will be maintained.

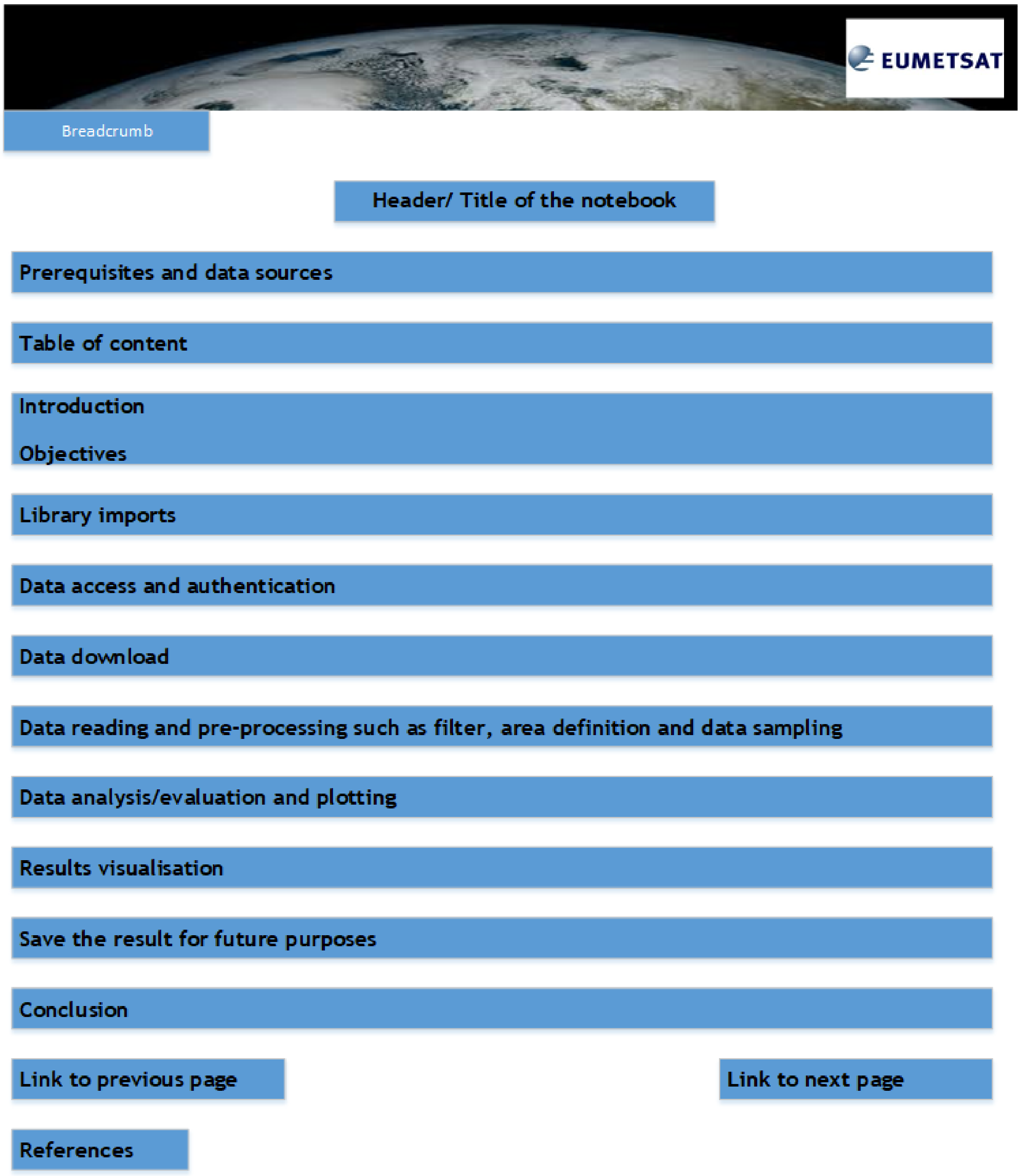

EUMETSAT
Breadcrumb
Header/ Title of the notebook
Prerequisites and data sources
Table of content
Introduction
Objectives
Library imports
Data access and authentication
Data download
Data reading and pre-processing such as filter, area definition and data sampling
Data analysis/evaluation and plotting
Results visualisation
Save the result for future purposes
Conclusion
Link to previous page
Link to next page
References

Figure 1 Layout of the theme for the notebooks

Chapter Three

Recommended Practices for Notebook Development

In addition to observing good coding practices, the following are recommended while developing the notebooks.

1. Paths: it is recommended to use relative paths in all sections of the files to be developed to reduce the chances of the end-user modifying the program.
2. The nonlinear workflow of Jupyter notebook cells: Jupyter notebooks have a feature that the cells can be executed out of order. Values of variables in the most current run cell is used for the subsequent to-be-run cells. This can create inconsistent final results when a cell is executed from variables that have not been updated in a previous and dependent cell. Thus, it is recommended to keep track of the dependent cells and update them to ensure consistent results. On the other hand, you can restart the kernel and or clear all outputs when developing to have a clean variable space and then rerun the entire cells to ensure consistency of values of variables.
3. Whenever possible, use the latest and stable version of any product package or library to ensure compatibility in the usage of the files by the end-users.
4. Avoid long comments in the code cells for explaining the code, rather make use of markdown cells for documentations and explanations to make the notebook more readable.
5. Notebook files need to be organised in directories according to the various clusters to make them easier to be found.
6. Notebooks have to be named using more descriptive names according to the theme under which it is developed.
7. Break longer segments of code into multiple logical cells or sections: Try to keep your code cells as short as possible. Break them up by adding markdown cells in between and add explanatory text. A cell for a single line of code is too short. A cell with more than 15 lines of code is too long.

8. Make sure to understand the needs of the target users and direct the development of the notebooks to directly influence their understanding of the theme. This will make it easier for them to use the notebooks and more likely recommend the notebooks to others.
9. After each significant transformation, show preview of downloaded data by printing the first three or five records of your DataFrame.
10. Commenting the code in the code cells: It will be useful to add comments to the code cells to explain its functionality if it is not immediately obvious. Avoid directly explaining what the line of the code does as this might be useful in the markdown text section. You can add comments within the code cells using the \`#\` symbol. To use the text as descriptive text, convert the cell to markdown cell and type only the text without the '#' symbol.
11. It will be very essential to organise sections of the code that are repeated in other sections of the program into functions and call the function in the appropriate section in which it is needed.
12. Have a section in the notebook to describe to the end-users the order of execution of the code cells so they will achieve the expected results to improve their learning. This can be done for instance in the introduction section.
13. Write clear and concise messages in attachment to push on the GitHub directory so other collaborators can understand the pattern of development of the programs. This can also serve as a guideline for them for collaborative editing.
14. Clearly indicate all the required dependencies required to run the notebooks successfully.
15. Use plain language to make the notebook easier to understand for all users. Define any technical term or not common item in a glossary or as a footnote.
16. If possible, provide contact details for a user to approach when he or she has some doubt or questions about the notebook (code or text content).
17. Clearly outline the objectives you intend to achieve at the dedicated section of the notebook. This sets the stage for the development of your notebook and helps the users understand the notebook's purpose.

18. Whenever possible, use dynamic values for variables.
19. Provide references to useful materials that will only be helpful to the end-users in understanding the contents of the notebook.
20. In view that the end-user might use the program inappropriately, develop the notebook code cells handling cases of errors and providing relevant error messages that include troubleshooting procedures and procedures to handle the errors.
21. It often makes sense to split the work into several notebooks for projects covering various topics or analyses. This way, each Notebook can stay focused and more readily understandable, making it easier to collaborate with others and revisit the work later on.
22. While developing the program, for plots:
    a. That involve subplots and parameters for more than a day, test for a large number of days to be satisfied with the appearance of the subplots. Otherwise, set a limit on the number of days for which the user can plot the parameter on the subplots. In whichever case, the user needs to be informed on the number of days required for the plots to make the program more user-friendly.
    b. Use dynamic values for the sizes of the texts on the plot, whenever possible.

Chapter Four

Markdown Elements

4.1 Use of Markdown Elements

A Jupyter notebook is a series of cells that can store code and descriptive text in a structured manner. Code cells allow you to write and execute programs while markdown cells allow you to write and render texts in already provided markdown elements. The notebooks to be developed are for both experts and non-experts thus they do not have to be only code but descriptive texts placed in various sections of the program to improve the readability and flow of the notebooks (especially the code) and to make it easier for end-users to understand the work process. This makes the notebooks artistically and visually appealing.

To use a cell as Markdown element, create a markdown cell by selecting the Markdown item from the dropdown menu shown in Figure 2 over the desired cell. The cell is converted to a markdown cell when the In [ ]: on the left of the line disappears.

After writing and formatting the text as desired, run the cell to see the formatted text.

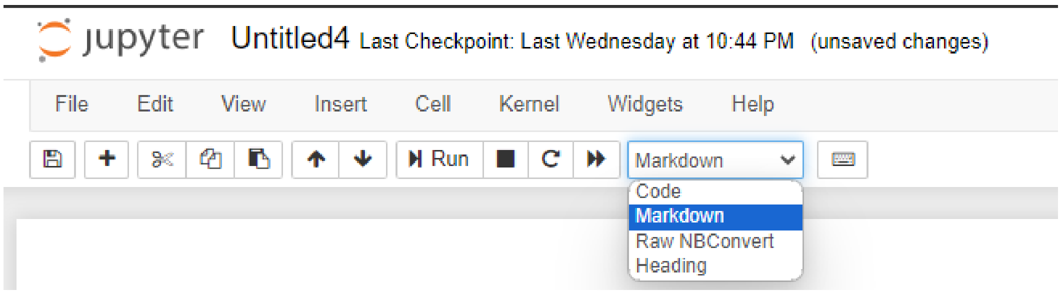


Figure 2 Dropdown menu containing markdown item to convert a cell to markdown cell

4.2 Some Markdown Elements

4.2.1 Abbreviations

To improve on the readability of the document, abbreviations could be defined as a tool tip using the HTML <abbr> tag. The definition for the abbreviated term shows whenever the user hovers over the term. The syntax to use is as follows:

<abbr title="Definition of the abbreviation"> the abbreviated term here </abbr>

4.2.2 Adding Widgets for User Interactions

Widgets are graphical user interface elements that have the potential to improve the interactivity of the user with the notebooks. Some of these elements are sliders, text boxes, checkboxes, etc. They can be used in conjunction with the code and markdown cells to allow the user to interact with the notebook. Here is a reference link on the documentation for ipywidgets in Jupyter notebooks.

4.2.3 Backslash Escape

You can use backslashes ('\') to escape characters that have special meaning (such as *, _, #) in the text of the markdown element. This prevents markdown from interpreting the character as an instruction rather than as the character itself. You can also escape a backslash with another backslash.

4.2.4 Block Quotes

To give highlight or emphasis to an information or a quotation in the notebook, you can use indented block quotes by preceding the text with a greater than symbol (>). This will set the emphasised text apart from the rest of the text.

4.2.5 Coloured Note Boxes

To give highlights to the contents of a cell, you can change the background colour of the cell using a combination of HTML and inline CSS styles. Table 1 gives some suggested colour types and the alert type they represent.

Table 1 Alert types

Box colour	Alert type
Blue box	Info
Yellow box	Warning
Green box	Success
Red box	danger

4.2.6 Collapsible Sections

To facilitate easy flow when reading the notebooks, it will be important to use this feature to hide/show details, whenever necessary, and present a summary to the user. The details are shown or hidden when the user clicks on the summary item. This functionality is typically achieved using the "details" and "summary" HTML elements. The "details" element creates a container for the detailed contents, and the "summary" element provides a clickable summary for the container. Note that the summary will always be shown in both instances of hide or show of the details. On the other hand, accordions could be used to create collapsible contents. More details on this behaviour could be found here.

4.2.7 File Attachments

To attach a file such as pdf, graphics etc. to the notebook, see the guide provided in Section 4.2.15.

4.2.8 Headings:

Please note that Jupyter notebooks no longer use heading cells and instead use the markdown cells for headings.

The notebook files have to be divided into sections and grouped under various heading levels for the topics and subtopics. A notebook should ideally have only one

Heading level 1 that is assigned to the theme or the main topic under discussion and then multiple headings of levels up to 6 nested under the main topic. To create a heading, use # or a number of it according to the number of heading level required as follows

```
# Header 1
  # Header 1.1
  # Header 1.2 …
## Header 2
### Header 3
#### Header 4
##### Header 5
###### Header 6
```

Note that for the headings to render correctly there has to be a space between the # symbol and the text.

4.2.9 Horizontal Line

This can be useful for separating various sections of the notebook to create horizontal line in the notebook, you can use three or more hyphens (---), asterisks (***), or underscores (___). Alternatively, you can use the HTML tag <hr>.

4.2.10 HTML Entities

HTML entities are sequences that begin with an ampersand (&) and end with a semicolon (;). They are created from a transformation of special characters such as ‘<’ into safer equivalents. The Table 2 shows some of these special characters and their safer equivalents recommended for use in the application developments.

Table 2 HTML Entities

Special Character	Safer Equivalent
"	&quot;
‘	&apos;
<	&lt;
>	&gt;
&	&amp;
©	&copy
®	&reg

For a complete list of the characters, please consult the page on this link.

4.2.11 Inline Code

To format text as inline code, you can use backticks (`) and if you want to show the back ticks as part of the inline code, you can use double ticks.

4.2.12 Insert Raw HTML

In Jupyter notebooks, you can insert HTML code directly into Markdown cells to add rich formatting, styling, or interactive elements that might not be achievable with pure Markdown. Here are some reasons why you might want to insert HTML in a Jupyter notebook:

1. HTML allows for more advanced formatting and styling options compared to Markdown. If you need precise control over the appearance of your content, using HTML can provide additional styling features.

2. HTML can include interactive elements like forms, buttons, or other JavaScript-based components. If you want to incorporate interactive elements into your notebook, HTML is a powerful tool.

3. HTML can be used to embed external content such as videos, iframes, or widgets from other websites or services.
4. If you have specific styling requirements that are not achievable with Markdown, you can use HTML for custom styling or layout.

To improve on the readability of the document, it is strongly recommended to limit the use of inline HTML code in the text of the markdown, considering that some of the users of the program might be coming from a non-programming background.

4.2.13 LaTex for Equations

LaTeX is a document preparation system for high quality type-setting, often used for technical or scientific documents. To give a more professional look to mathematical equations used in the preparation of the notebooks, it is recommended to use the LaTeX tools. It has its own syntax for rendering mathematical equations in the document. For more information on the syntax for writing various mathematical equations, please visit the site in this link. To display an equation within the text of the notebook, use a single dollar sign. On the other hand, to display an equation on its own line, use a double dollar sign or the syntax below:

```
\begin{equation}
    Your mathematical expression
\end{equation}
```

4.2.14 Line Break

A line break is the termination of a line and continuation of text writing on a new line. To create a line break, end a line in the markdown cell with 2 or more spaces at the end of the line or insert <br> at the line termination point.

4.2.15 Links

You can create links using square brackets for the link text and parentheses for the target URL. The text in the square bracket is the text that will be displayed in the notebooks. This is the basic syntax for creating hyperlinks.

```
[link_text](target_url)
```

when the user clicks on the link text, he will be redirected to the target URL.

The following are some diverse ways to provide navigation links in Jupyter notebook. To provide a link to another notebook, use the following syntax:

```
[link_text](relative_path_to_notebook)
```

You can also insert an image in the notebook and make it clickable to another link using the syntax below:

```
[![alt_text](image_path)](url_to_redirect_to)
```

where ‘alt_text’ refers to the text that is displayed when the image is not available.

Anchor Links

Anchor links are links that redirect to sections within the same notebook page. It is useful for easy navigation within long notebooks. To create anchor link, use the following syntax. Replace white spaces in section labels with a hyphen and also escape special characters with a backslash in order for the link to work well.

```
[link_text](#section_label)
```

To add a mouse over title to the link, use the following syntax:

```
[link_text](target_url “mouse_over_text”)
```

Alternatively, to display the text of the target URL directly in the text of the markdown, you can use the syntax:

```
<target_url>
```

Ensure the link labels are unique to avoid misdirection in the document. Also, know that the link labels are case sensitive.

References Style Links

To keep the contents of the cells more readable, you can use reference-style links to separate the link definition from the link text. The syntax is as follows

[link_text][link_label]

Using the [link_label], the reference is defined as follows

[link_label]: target_url “optional_mouse_over_text”

References and Citation

Place the citation label (consisting of text surrounded by square brackets) in the text to be referenced.

In the reference section, provide the details of the citation using the corresponding citation label.

[^citation_label]: Author name, "Title of the article", Name of journal, year.

where *citation_label* is the label used for the citation in the text of the document. Note that the reference will appear as a footnote when the document is converted to pdf. For a more advanced referencing style, consult this page.

4.2.16 List

The following are the ways of creating lists in Jupyter notebook. Note that you can also create a list of two or more of the types described below.

Ordered List

To create an ordered list, use numbers followed by a period and a space before the text of the list items. You can use any sequence of numbering and markdown will automatically adjust the sequences based on the starting number of the list in the cell.

Unordered List

To create an unordered list, you can use the asterisk `*`, plus `+`, or minus `-` sign followed by a space before the text of the list items.

To create a sub item in both ordered and unordered list items, use a tab before initiating the list item.

Checked Lists

You can create lists using checkboxes. To do so, you have to use square brackets with or without an ‘x’ to mark the status of the checkbox. In this case, when creating a list of tasks that have to be completed by the user, the checkbox will be helpful to guide the user to check the number of tasks that are required to be completed. A guide message will thus be very useful in the application on how the user can change the status of the checkbox.

4.2.17 Mentions

You can make reference to other collaborators working on a project using the ‘@’ symbol followed by their username.

4.2.18 Paragraphs

The text of the document can be separated into paragraphs to make them more readable. To do so, simply insert a blank line between the paragraphs.

4.2.19 PDF Convert

To convert your Jupyter notebook files to PDF, it is advised to use the terminal. Proceed as follows:

i. open the terminal
ii. navigate to the directory of your Jupyter notebook file
iii. run the following command

```
Jupyter nbconvert --to pdf file_name.ipynb
```

iv. when successful, a pdf file with the same name as your Jupyter notebook file will be created in the same directory as your notebook file.

To control the layout of the pages when converting the file to pdf, you can use page breaks to start on a new page. To do so,

i. create a markdown cell at the section where you want to have the page break
ii. type

```
\pagebreak
```

iii. Convert the file to pdf using the procedure described above.

You can also attach a pdf file to your notebook file for reference or to be downloaded by the user using the following step:

```
from IPython.display import display, IFrame
pdf_path = 'path_to_your_file'
display(IFrame(pdf_path, width=width, height=height, allowfullscreen=True))
```

To give only a link to the pdf file, see the section on file attachments/links.

4.2.20 Strikethrough

In a situation in which a model or function is no longer applicable under the theme of discussion, strikethrough feature could be used to highlight that the model or function is no longer valid. To do this in the markdown cell, surround the text of the model with a tilde (~), ensuring that there is no space between the tilde and the text. For example:

```
~an outdated model~
```

4.2.21 Subscript and Superscript

You can create subscript or superscript by using the latex format or the HMTL tags shown below

	Superscript	**Subscript**
Latex	$^{superscript_text}$	$_{subscript_text}$
HTML syntax	<sup>superscript_text </sup>	<sup>superscript_text </sup>

4.2.22 Syntax Highlighting

To demonstrate a code text in the descriptive text, you can improve the readability of the code and the entire text by using syntax highlighting. For inline code, you can use 3 backticks to enclose the code and also specify the programming language after the opening backticks.

4.2.23 Table of Contents

A table of contents is very useful to allow easy navigation within the notebook and most especially, when the document is converted to a pdf file. This can also be used to create a navigation menu for different sections of the notebook.

To create a Table of Contents in the document, you can use anchor links as follows.

1. Create sections in the document using headings of appropriate levels
2. Add anchor links beneath each section heading
3. Create a new markdown cell at the beginning of the document, where you want to have the table of content and populate it with the reference links to the various sections and subsections in the document
4. Update the table of content as and when you add a new section to the document or modify the existing sections and subsections.

4.2.24 Tables

You can create tables in the notebooks using pipes (|) and hyphens (-). The pipes are used to create the columns on the same line and on successive lines to create the rows of the tables as shown in the syntax below.

```
| Header 1 | Header 2 | Header 3 |

|:---------:|:--------:|:--------:|

| Content 1 | Content 2 | Content 3 |

| Content 4 | Content 5 | Content 6 |
```

The spacing between the pipes does not matter as they will be automatically aligned when the markdown cell is rendered. The second line is very useful to differentiate between the header line and the cell contents.

You can also change the alignment of the contents of the cells using a combination of colon and hyphens as shown in the table below.

Pattern	Alignment
:----:	Centre
----:	Right
:----	Left

Note that you can also use HTML tags to create tables in Jupyter notebooks.

4.2.25 Text Formatting

You can use HTML tags and inline CSS style attributes 'color' and 'font-family' to respectively change the colour and font of a text, by specifying values for these attributes.

Also, you can surround the text with double asterisks (**) or double underscore (__) to bolden the text or the single equivalents to italicise the text.

Lastly, you can change the alignment of the text using HTML tags and inline CSS styling '*text-align: type*' with the type values of 'justify', 'center', 'right' or 'left'.

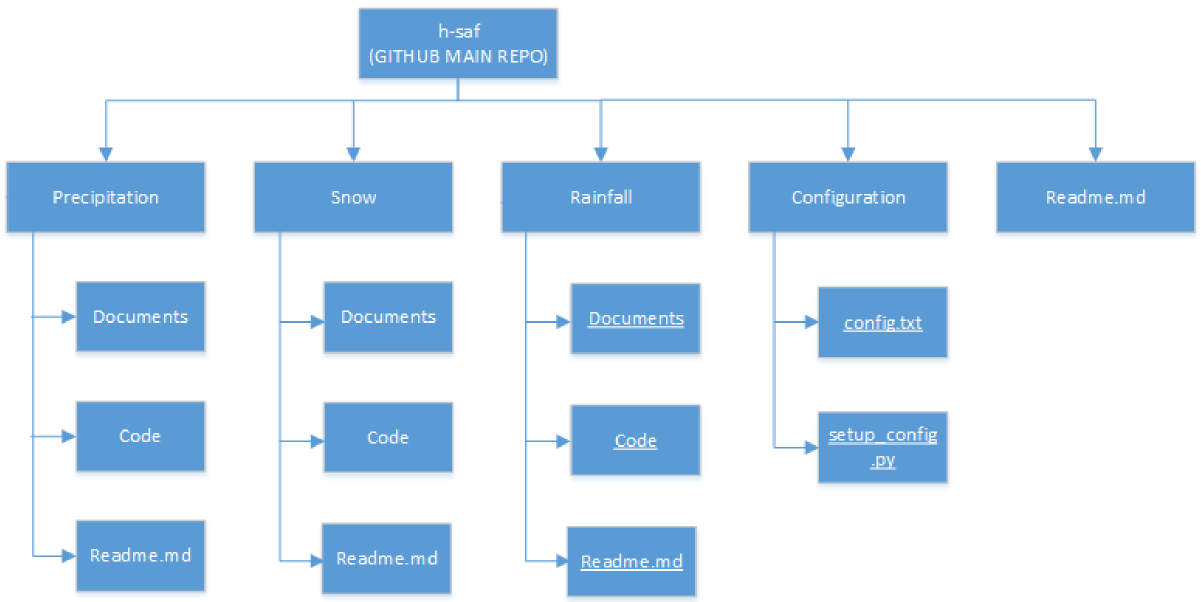
Chapter Five

GitHub Repository Structure

To facilitate collaboration with other team members, it is proposed for the projects to be developed under a common umbrella. Thus, a parent folder, h-saf, having subfolders for each of the 3 clusters and for the program configuration files. In addition, a Readme.md file is required for the GitHub repository.

Under the folders for each of the clusters are subfolders documents and code folders and a readme.md file that gives description to the particular cluster in consideration. In the documents folder are to be located all the documentations that will be useful for the end-users and meant to be made available to them. In the code folder is to be located the programs for utilising the h-saf data products also organised into exercises and lessons folders.

In the configuration folder are to be located 2 main files, a config.txt and setup_config.py files. The config.txt file is to contain the list of all dependent programs or libraries that will be required for running the programs while the setup_config.py file is to be used for creating the virtual environment and install the required dependencies for running the programs.



Chapter Six

Conclusion

6.1 Conclusion

To promote usage of the rich products of H-SAF, there is a need to provide a guide for the development of materials that will be very useful for user training and access to the data products. This guide thus seeks to provide tools for the development of computational notebooks that will be educative, user friendly and instructive to the end-users. A template have been developed using this guide and it is anticipated users will find it very useful accessing the H-SAF data products.

As future updates to the template tools and the associated scripts will be expected, it is recommended for the changes to be documented in this document, and the version of the document updated, to reflect the changes made.

References

Wagemann, J., Fierli, F., Mantovani, S., Siemen, S., Seeger, B., Bendix, J., (2022) 'Five Guiding Principles to Make Jupyter Notebooks Fit for Earth Observation Data Education.', *Remote Sensing*, 14, 3359, https://doi.org/10.3390/rs14143359.

Websites referenced in the document:

- https://ipywidgets.readthedocs.io/en/stable/
- https://www.w3schools.com/howto/howto_js_accordion.asp
- https://mateam.net/html-escape-characters/
- https://en.wikibooks.org/wiki/LaTeX/Mathematics
- https://jupyterbook.org/en/stable/content/citations.html#:~:text=You%20can%20add%20citations%20and,with%20the%20%7Bbibliography%7D%20directive.