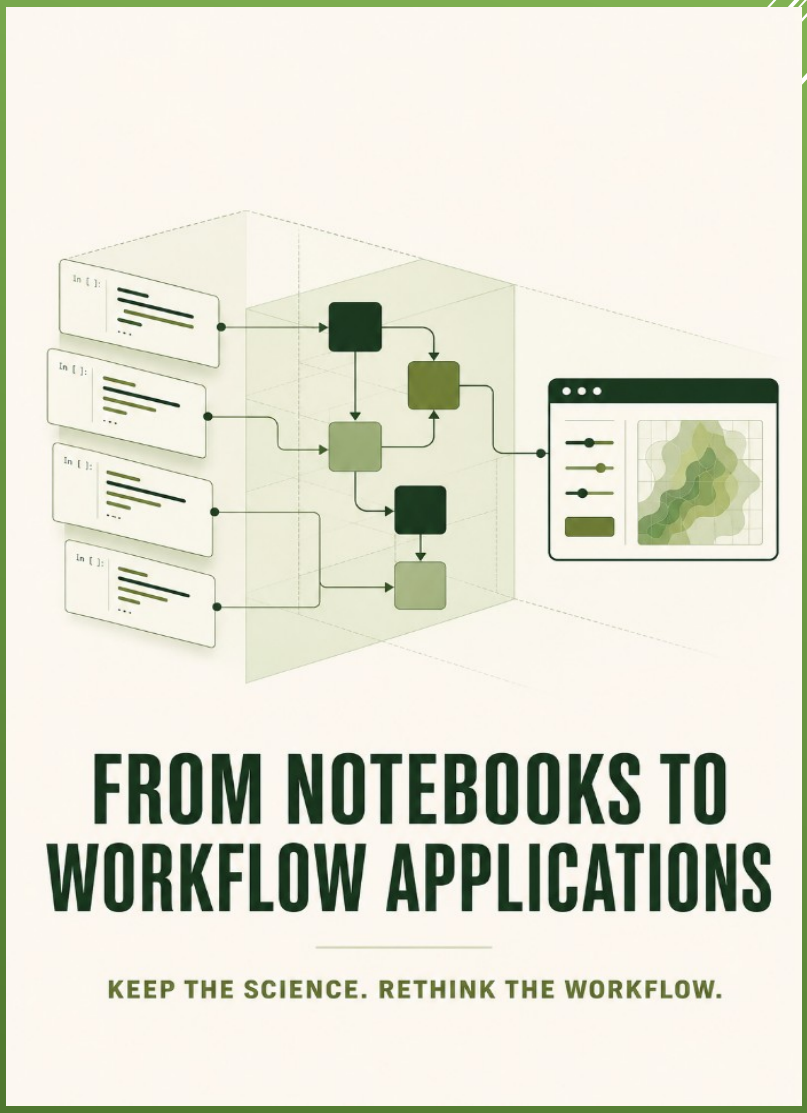
Clim EMPOWER



TRANSFORMING JUPYTER NOTEBOOKS TO OPERATIONAL APPLICATIONS

Part of ClimEmpower educational library

Project information	
Project name:	Project ClimEmpower: User Driven Climate Applications Empowering Regional Resilience
Grant Agreement No.	101112728
Start date:	01/09/2023
Duration:	36 months
Coordinator:	Denis Havlik, Scientist <denis.havlik@ait.ac.at> AIT Austrian Institute of Technology Giefinggasse 4, 1210 Vienna, Austria
	Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.
Deliverable details	
Report name	**Transforming Jupyter Notebooks To Operational Applications: A Rapid Prototyping And Co-Creation Guide**
Target audience	This guide is targeted towards Data Scientists and Software Engineers
Description:	Practical guidance on transforming Jupyter notebook workflows into accessible applications that make scientific analyses easier to use, understand and share, part of ClimEmpower educational library
Estimated reading time	~30 minutes
Learning level	Intermediate
Prerequisites	Basic understanding of software development, Python, Python geo-data related libraries, Jupyter notebooks, Javascript and AI-assisted development
Keywords	ClimEmpower, educational report, Jupyter notebooks, Workflow applications, LLM, Voila, Climate services, Flood Risk, Flood Hazard
Downloadable from	• ClimEmpower web site: https://climempower.eu/ • Zenodo: 10.5281/zenodo.21715952
Version: **Publication date:**	**V1** **10/09/2026**
Dissemination level:	PU = Public
Editor:	Havlik, Denis - AIT
Author(s):	Kunhanam Veedu, R., Skuta, A., Havlik, D. (AIT Austrian Institute of technology GmbH.), Austria

Recommended citation

Kunhanam Veedu, R., Skuta, A., Havlik, D., (2026). Transforming Jupyter notebooks to operational applications: A rapid prototyping and co-creation Gude, in ClimEmpower educational library, ed. Havlik, D., URI: https://climempower.eu/edu/, DOI: 10.5281/zenodo.21715952

Revision history

Date	Version	Contributor/Reviewer	Description
10/09/2026	Initial release	Rishana Kunhanam Veedu, main author Adam Skuta, co-author Denis Havlik, review and improvements	First public version of the report uploaded to Zenodo, doi reserved

Legal Disclaimer

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
CIC	Climate interaction context
AI	Artificial Intelligence
API	Application Programming Interface
JRC	Joint Research Centre
JSON	JavaScript Object Notation
LLM	Large Language Model

LUISA	Land Use-based Integrated Sustainability Assessment
NetCDF	Network Common Data Form

Glossary

Term	Definition
Climate impacts	The consequences of realized risks on natural and human systems, where risks result from the interactions of climate-related hazards (including extreme weather and climate events), exposure, and vulnerability. Impacts generally refer to effects on lives; livelihoods; health and well-being; ecosystems and species; economic, social and cultural assets; services (including ecosystem services); and infrastructure
Application state	Information retained about selected inputs, processing progress and available results.
Backend	The part of an application that performs processing and manages data behind the interface.
Deterministic validation	Checks based on predefined rules, used here to verify the analyser's response structure and references without confirming its scientific interpretation.
Frontend	The interface through which users provide inputs, start actions and view results.
GeoTIFF	A raster file format that stores geographic information alongside values such as flood depth.
Handoff	The transfer of an output from one workflow stage or notebook for use as an input by another.
Kernel	The process that executes notebook code and retains variables during a session.
Raster	Geographic data represented as a grid of cells, each containing a value such as flood depth or land-use class.
Return period	The average interval between events of a specified magnitude or greater; a 100-year event has a 1% chance of being equalled or exceeded in any given year.
Workflow application	An application that organises a processing workflow into explicit user inputs, actions, stages and results.

About this document

This document is part of ClimEmpower educational library and associated with project deliverable D4.2 “Educational materials for increased regional Climate Change resilience v2”.

*ClimEmpower’s key ambition is to **prove beyond doubt that CC-resilience should, and can, be an integral part of regional development** everywhere in EU and beyond it. That is, we anticipate that the regional stakeholders will recognise that CC-resilient development pathways offer multiple benefits to them, including but not limited to **higher quality of life and reviving economy**, and that these can be understood using available **data, tools, and services**. Second key ambition of the project is to **help the regions address the CC resilience** in key community systems addressed in five ClimEmpower trials.*

*Underlying philosophy of the project is to “**help the regions to help themselves**”. This will be achieved through various mechanisms, including co-creation and mediation of the regional “Communities of Practice”, provision of the Climate Change -resilience training materials, as well as in provision and training in use of the user-centric data and services – including those that have already been made available through previous research projects and EU initiatives.*

This manual is part of the ClimEmpower educational library. Available at https://zenodo.org/uploads/21715952

This material helps Engineers and Data scientists understand workflows on an efficient transformation from Jupyter notebooks to workflow applications and enables them to to rapidly develop such applications.

After reading this material, readers will be able to:

- **Understand** the key concepts, challenges, and principles related to Jupyter notebooks and workflow applications.
- **Explain** how transformations and libraries work and why it is relevant for rapid prototyping.
- **Apply** transformation methods on their own Jupyter notebooks in their own context to rapid prototyping.

1 Introduction

1.1 Why should I read this document?

This document provides a practical introduction to transforming Jupyter notebooks into presentable, explainable workflow applications for end users and non-technical audiences. It examines three approaches: enhancing notebooks directly with Voilà, converting them into dedicated backend–frontend applications using templates, and using agentic skills to automate parts of the transformation process.

If you develop notebook-based analyses, models, or workflows, this document will help you evaluate which approach best suits your use case. It explains how the different approaches can improve usability and presentation, what level of development effort they require, and how they support the transition from technical prototypes to accessible applications.

This material is particularly relevant to data scientists, researchers, software developers, and digital innovation teams who want to make notebook-based work easier to use, understand, share, and maintain. It can support decisions about application architecture, transformation workflows, automation opportunities, and the balance between rapid development and a tailored end-user experience.

1.2 The Challenge

Jupyter notebooks are widely used to develop analyses, models, and data-driven workflows, but they are primarily designed for technical users. Their combination of code, inputs, and outputs can make them difficult for non-technical users to understand and operate. As notebook-based solutions increasingly move beyond experimentation and are shared with decision-makers, domain experts, and other end users, the need for accessible and presentable interfaces is becoming more important.

Without an effective way to transform notebooks into user-oriented applications, valuable workflows may remain dependent on their original developers, require repeated manual support, or fail to achieve wider adoption. Development teams may also spend significant time rebuilding working prototypes as standalone applications.

This document addresses the challenge of turning notebook-based workflows into usable and explainable applications while balancing development effort, flexibility, maintainability, and the needs of end users.

1.3 What will I learn?

This document introduces three approaches for transforming Jupyter notebooks into presentable and explainable workflow applications. It explains how Voilà can be used to enhance and publish notebooks directly, how notebook workflows can be restructured into dedicated backend and frontend components using templates, and how an agentic skill can automate parts of this transformation process.

After reading this document, you will:

- understand the main challenges involved in turning technical notebooks into applications for non-technical users,
- know how Voilà, dedicated backend–frontend structures, and agentic automation can support this transformation,
- understand the benefits, limitations, and development effort associated with each approach,
- be able to compare the approaches based on factors such as usability, flexibility, maintainability, and automation potential,
- be able to select and apply an appropriate approach for transforming your own notebook-based workflows into more accessible end-user applications.

2 Concepts, methods, and approaches

Making a notebook workflow useful to a wider audience involves clarifying how users provide inputs, carry out processing and interpret results. This section explores how that can be achieved by enhancing the notebook itself with Voilà or manually translating its workflow into a dedicated application. It then introduces AI-assisted notebook analysis as a way to support the understanding and planning required for manual translation. Examples from a river-flood workflow illustrate how these approaches connect scientific processing with user interaction.

2.1 Enhancing Jupyter Notebooks

A Jupyter notebook is an effective prototyping format for data scientists and engineers because it combines executable code, explanatory text, visualisations, and intermediate results in one place. As a prototype matures, however, its cell-by-cell interface can make the workflow difficult to share and use consistently. Users must know which parameters to edit, which cells to run, and in what order.

Voila provides a fast route from notebook prototype to workflow application. It renders the notebook's Markdown, outputs, and interactive widgets in a browser while hiding the implementation code. Data scientists and engineers can therefore reuse the existing Python methods, documentation, and visualisations instead of rebuilding the workflow with a separate frontend and backend. The main conversion task is to organise the existing notebook logic into clear controls, actions, workflow stages, state transitions, file handoffs, and user-facing feedback.

2.1.1 From prototype to workflow application

Voila executes a notebook when a user session starts. A workflow application should use this startup phase to initialise the interface, not to begin workflow steps before the user has selected inputs and reviewed the settings. The conversion therefore separates application setup from user-initiated actions.

Downloads, raster processing, file writes, and damage calculations occur only after the user presses the corresponding action button. Simple interface interactions, such as copying an area preset into coordinate controls, may remain reactive. This separation gives users direct control and makes the stages of the workflow explicit.

The conversion maps common notebook patterns to application patterns as follows:

Table 1 Mapping table, that maps notebook patterns to application patterns

Linear notebook pattern	Voila application pattern
Edit a Python variable	Select or enter a value in a validated widget
Run the next cell	Press a clearly named action button
Depend on variables left in the kernel	Load validated application state or a saved product
Watch terminal-style output	Use progress indicators and dedicated status panels
Pause to edit a file	Download, edit, upload, and validate the file explicitly
Rerun cells that may overwrite outputs	Preserve user files and replace generated products safely
Infer whether results are current	Record input and result manifests and reject stale outputs

2.1.2 Case study: the river-flood workflow

The original workflow consisted of two connected notebooks: one prepared river-flood hazard maps, and the other combined those maps with land-use and economic information to estimate potential damage. Their scientific methods and file formats were retained, but their editable parameters and processing cells were reorganised into two guided applications.

The applications use tabs from left to right to communicate the intended sequence. They are not linear, most later buttons remain available, but each callback checks its prerequisites and reports a recoverable error if an earlier product is missing. This preserves flexibility for data scientists and engineers while giving other workflow users a clear path through the application.

2.1.2.1 Hazard application

The hazard application has three stages:

1) **Define the area**. The user selects a preset or enters an area name and WGS84 bounding box. Names and coordinates are validated before they are used in paths or geospatial processing.

2) **Prepare and review present-day hazard**. The user selects JRC return periods and explicitly prepares the flood maps. Plotting is a separate action, allowing prepared maps to be viewed with different display settings without repeating downloads or raster processing.

3) **Prepare and review climate context**. The user selects an Aqueduct return period, climate scenarios, years, and models. The application prepares a historical baseline and future model ensemble, then plots future flood depth and the difference from the baseline. These coarser Aqueduct results provide

qualitative climate context rather than direct inputs to the economic damage calculation.

The application clips remote source rasters to the selected area, reprojects them to EPSG:3035, and stores area-specific GeoTIFF, NetCDF, configuration, and plot files. Product names remain compatible with the downstream risk application. Preparation and plotting are deliberately separate so that users can change presentation settings without repeating an earlier workflow stage.

2.1.2.2 Risk application

The risk application uses the prepared hazard products through five stages:

1) **Load.** Discover an area prepared by the hazard application, validate its JRC NetCDF, and load the available return periods.
2) **Review.** Plot present-day JRC hazard and, when available, the optional Aqueduct climate context.
3) **Exposure.** Download or reuse LUISA land-use data, clip it to the hazard extent, and align the selected flood maps to the land-use grid. Create an area-specific economic workbook only if one does not already exist.
4) **Economics.** Let the user download and edit the workbook in a spreadsheet application, then upload it through the browser. Validate its land-use classes, required columns, values, and building-type ratios before replacing the saved workbook.
5) **Results.** Run DamageScanner for the prepared return periods, display category-level damage totals, and produce a three-panel map of potential damage, flood depth, and land use.

The handoff between applications is file-based rather than dependent on a shared notebook kernel. The risk application discovers the NetCDF written by the hazard application and checks that it has a coordinate reference system, spatial dimensions, and return-period variables before accepting it. This makes the dependency visible and allows the applications to run in separate sessions.

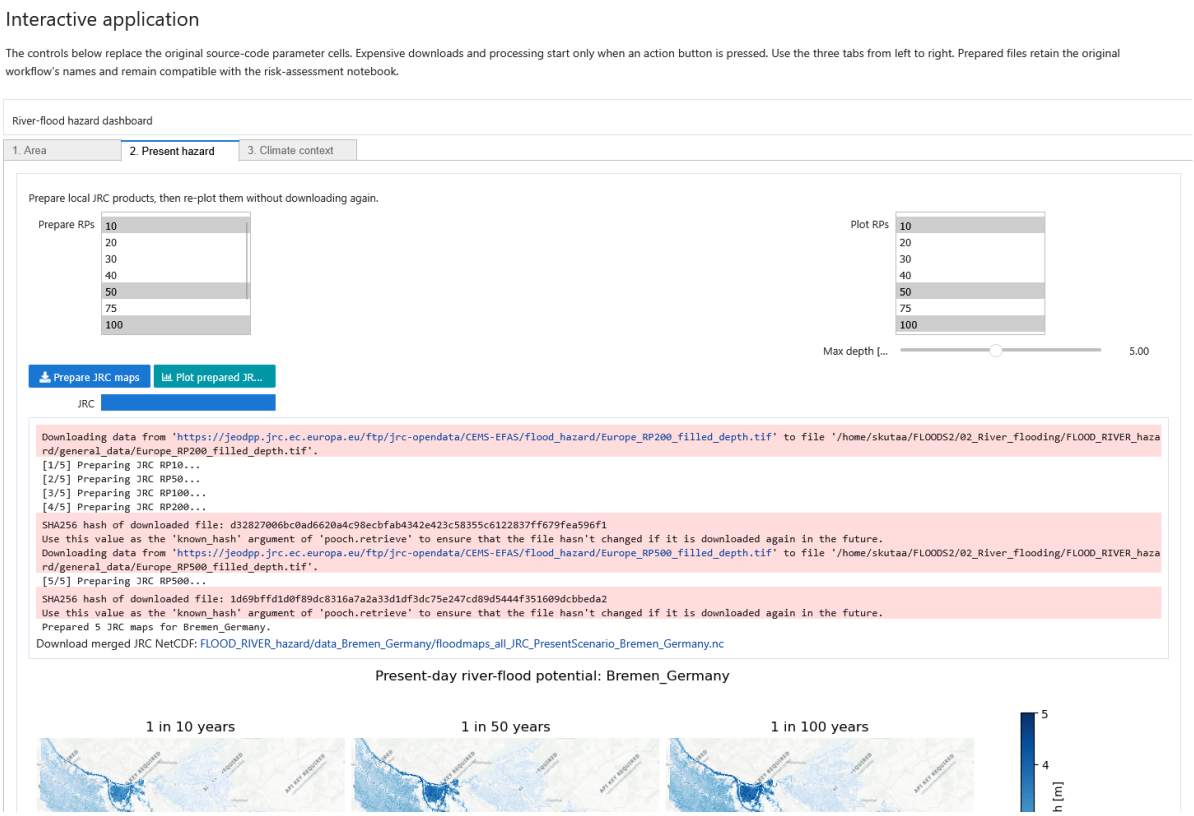


Figure 1 Hazard workflow represented as a Voila application

2.1.3 Scope and limitations

This approach prioritizes conversion speed and reuse of an existing notebook over complete freedom in application design. A Voila application built with *ipywidgets* is not as visually clear, polished, or adaptable as a purpose-built application with a designed frontend and backend. A handwritten web application provides greater control over navigation, responsive layout, visual hierarchy, accessibility, client-side interaction, session management, and deployment architecture. It also requires additional engineering work and may duplicate logic or visualizations that already exist in the notebook.

The conversion approach described here has only been implemented and tested on the connected river-flood hazard-to-risk workflows presented in this case study. It is therefore centered on their structure: sequential scientific processing, geospatial datasets, action-driven preparation stages, file-based transfer from a hazard application to a risk application, editable economic input, and server-rendered results. The general principles may be useful for other notebooks, but they have not yet been validated across unrelated workflow types. Data scientists and engineers should adapt the stages, state model, validation rules, and interface controls to their own application rather than treating this case study as a universal conversion template.

2.2 Manual notebook to application translation

Jupyter notebooks are mainly designed for the purpose of exploration and technical development. A Jupyter notebook's workflow can be influenced by the order of the cells, the editable variables, the need to inspect intermediate results, and the values that are kept in an active kernel. While these features are helpful when carrying out an analysis, they do not always offer a clear or reliable experience to users who are applying the software.

When manually translating a notebook into an application, the notebook is treated as evidence of the underlying workflow rather than as the final application design. The engineer then reconstructs the intended process, determines which parts should be kept, and rearranges these into application components. The aim is not to replicate each and every cell, but to retain the validated scientific workflow while making its inputs, the various processing stages, its dependencies and its outputs clear.

Although this method involves a greater amount of engineering work than simply improving the notebook, it does offer more control with regard to the application architecture, the interface, validation, state management and the user experience.

2.2.1 From notebook elements to application elements

Notebook and application structures serve different purposes. Translation therefore requires interpretation rather than a direct one-to-one conversion.

Table 2 Typical mapping of notebook elements to application components

Notebook element	Application interpretation
Editable variable	User control or internal configuration
Processing cell	Reusable function, module or backend operation
Sequence of related cells	Meaningful application stage
Cell output	Map, table, message, file or other user-facing result
Intermediate variable	Explicit application state or persisted product
File created for another notebook	Defined handoff between workflow stages
Markdown instruction	Interface guidance, label or supporting documentation
Manual action outside the notebook	Explicit user task or application interaction
Exploratory or debugging cell	Usually excluded unless it provides user value

The purpose of the application determines the mapping; for instance, a value that is frequently edited by users can be made into an interface control, whereas a technical value which should stay stable will be retained as internal configuration. It is also possible to combine several notebook cells into a single application stage when they together carry out one user task.

2.2.2 A practical translation process

A notebook workflow can be translated through the following activities:

1) Define the purpose and identify the users; find out what the workflow is meant to achieve, who will be using the application and what results they need.
2) Work out the workflow by reading both the code and the Markdown to determine the needed inputs, the configuration, the key processing steps, the intermediate products, the final outputs, and the dependencies.
3) Separate out the work that can be reused from that which is for exploration. Make a distinction between the operations that are needed for repeated use and those cells that are used for experimentation, debugging or for intermediate inspection.
4) Establish the dependencies and handoffs. Find out what each stage takes in and gives out. Once the notebooks are connected, identify the files or data that are transferred between them.
5) Carry out processing separately from interaction. Arrange the reusable scientific operations independently of the interface controls, the actions, the messages and the way they are presented.
6) Plan the workflow for the user by grouping related operations into clear stages and by replacing editable variables and implicit cell execution with the appropriate controls and named actions.
7) Make the state of the workflow explicit by specifying how the application stores the intermediate results, checks the prerequisites, and deals with missing or invalid inputs without depending on an active notebook kernel.
8) Check the translation by comparing the application's inputs, processing steps, dependencies and outputs with those in the notebooks to make sure that the intended scientific meaning has been kept.

The process isn't entirely mechanical since verification can show that the workflow boundaries, dependencies or the suggested controls will have to be looked at again.

2.2.3 Decisions requiring engineering judgement

Certain translation choices are determined by the intended users and the operating environment. The engineer has to decide:

- which notebook values users should be allowed to change;
- which values should remain internal configuration;
- how the processing should be divided into application stages;

- which intermediate products need to be stored;
- how prerequisites and invalid inputs should be handled;
- which notebook visualisations are useful to application users;
- whether manual actions should remain manual or become application functionality; and
- which exploratory operations should be omitted.

This is the reason why a notebook cannot be turned into a dedicated application by means of a completely deterministic process carried out on a cell-by-cell basis.

2.2.4 Use case: translating the river-flood workflow

The manual-translation experiment used the same two connected river-flood notebooks as the Voilà-based enhancement. Instead of retaining the notebooks as the application structure, their underlying workflow was reorganised into a dedicated application with separate processing steps and interface components. This allowed the hazard and risk stages and the handoff between them to be represented explicitly. Using the same source notebooks demonstrates how the two transformation approaches represent the same scientific workflow differently.

2.2.4.1 Translating the hazard assessment

The hazard notebook includes the procedures needed for defining an area, for retrieving flood data, for clipping and reprojecting rasters, for processing optional climate information and for displaying the results.

In the application:

- Area selection became a choice between a configured region, a drawn bounding box, and an uploaded shapefile
- The JRC return period became a user control
- Optional Aqueduct scenario, year, and return-period values became climate-comparison controls. Instead of being triggered implicitly through notebook execution, the climate comparison is presented as a separate application tab with its own controls and action
- Notebook processing became a named action for generating the primary hazard map; and
- The resulting rasters were then presented as map layers in the interface.

When the application is asked to save an area-specific JRC GeoTIFF, it also generates the Aqueduct baseline, future ensemble-mean and flood-depth change rasters.

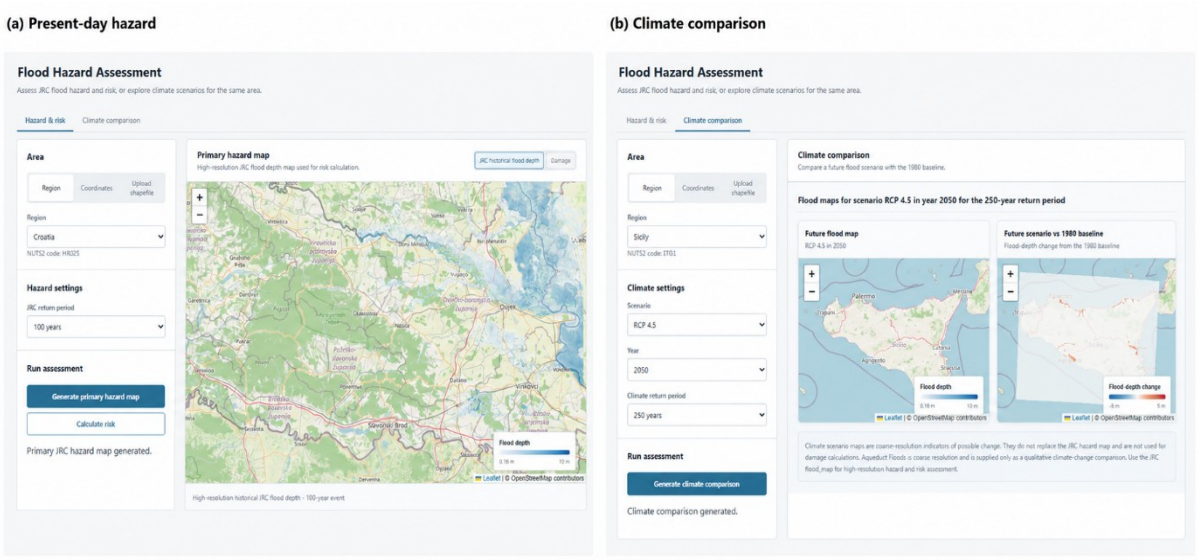


Figure 2 Present-day hazard assessment and the separate climate-comparison tab

2.2.4.2 Translating the risk assessment

The risk notebook includes prepared hazard information together with LUISA land-use data, damage curves and economic information.

In the application:

- Risk calculation becomes available after the primary JRC hazard raster has been generated.
- The saved hazard raster forms the input to the risk stage.
- The hazard and land-use datasets are aligned.
- Configured damage information supports the DamageScanner calculation.
- The resulting damage is shown as a map layer.

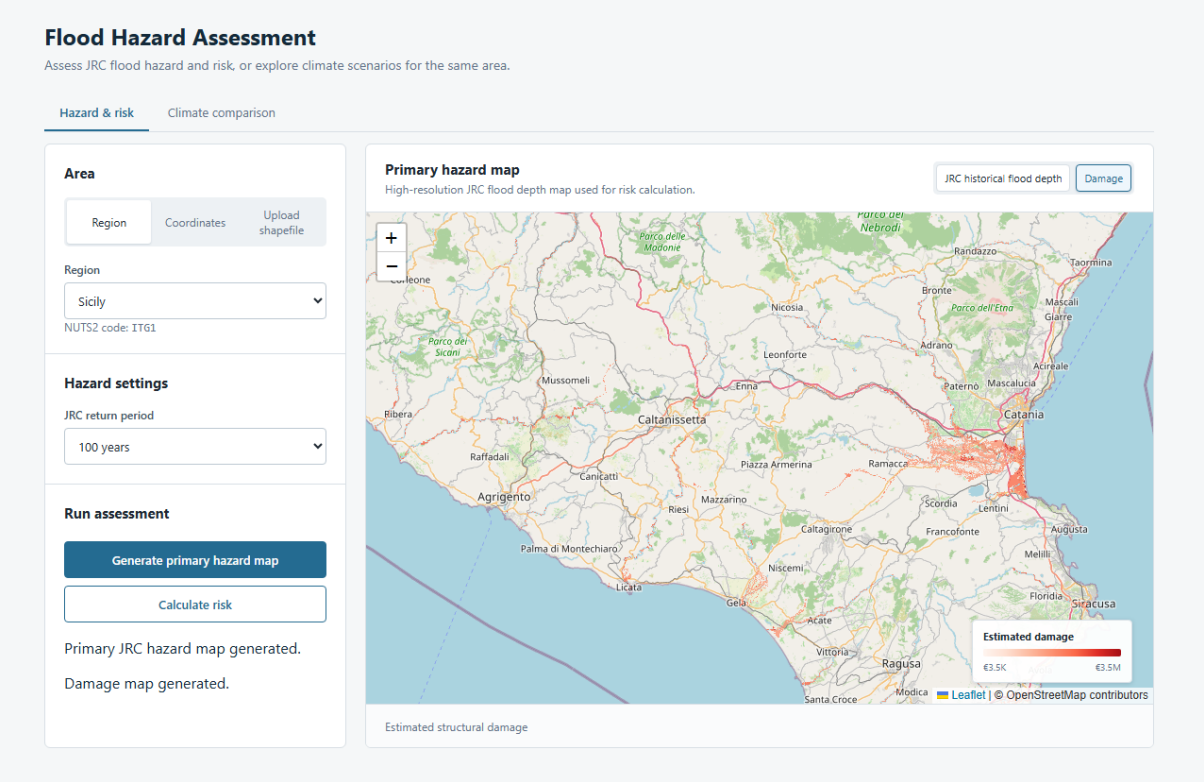


Figure 3 Notebook analyser interface with notebooks prepared for submission

2.2.4.3 Making the notebook relationship explicit

In the notebook environment, subsequent processing might rely on variables, paths or files that are known to the notebook author. The application substitutes this implicit dependency with a clear handoff.

The risk operation is given the name of the area and the path to the hazard file by the frontend. Before using the raster, the backend checks that the path is still within the specified output directory and that the file actually exists. This ensures that the risk calculation is not initiated through the interface until a hazard product is available.

2.2.4.4 Preserving scientific meaning

A decision had to be made about which notebook parameters users should control through the application. In the notebooks, the assessment area and return periods are defined by editing variables in code cells. In the application, these values are represented through explicit controls. Users can select a configured region, draw a bounding box or upload a shapefile, choose a JRC return period and optionally configure a climate comparison.

Other technical values remain part of the internal processing. This demonstrates that manual translation requires engineers to distinguish meaningful user choices from implementation-specific configuration rather than exposing every editable notebook variable through the interface.

2.2.5 Scope and limitations

The example involving flooding shows how manual translation can be carried out in a sequential geospatial workflow which includes both hazard and risk stages. The architecture and the controls of this approach are designed to meet the needs of these notebooks and must not be regarded as a general-purpose application template.

Some workflow types might need different component boundaries, state management, validation, storage, authentication, background processing or interface patterns. The fact that a translation has been carried out does not in itself guarantee that it is scientifically correct. The processed outputs of the translation have to be checked against the original workflow, and the relevant domain experts must confirm that the intended scientific meaning has been kept.

2.3 AI-assisted notebook analysis for manual translation

Translating a Jupyter notebook manually into an application can take up a lot of time and effort, especially if the notebooks are long, unfamiliar, exploratory or spread over several files. Before any application design can start, the engineer has to look at the code and the Markdown, work out the intended workflow and spot any reusable processing, inputs, outputs and dependencies.

An initial structured interpretation of the workflow can be provided by means of AI-assisted notebook analysis, giving the engineer a useful starting point without taking the judgement and verification that are necessary during manual translation.

2.3.1 AI support for understanding notebook workflows

An AI assisted analyser looks at the content available in the notebook and makes an effort to reconstruct the reusable workflow that it represents. It may combine an LLM with deterministic rules or static-analysis tools. The general technique is thus not dependent on a specific model, prompt, API or infrastructure.

A useful analyser considers code and Markdown together and may identify;

- The overall purpose and intended outcome
- External inputs and behaviour-changing configuration;
- Meaningful processing stages and their order;
- Intermediate and final outputs;
- Dependencies and handoffs between stages or notebooks;
- Manual actions and exploratory work;

- Explicitly used libraries, datasets and services; and
- Questions which cannot be settled on the basis of the available evidence.

What we want to achieve is not to draw up a single step for each cell; instead, several cells may combine to form one meaningful operation, while others may have the role of supporting exploration or presentation. An effective analysis should describe the workflow at an engineering level and at the same time keep connections with the original evidence.

2.3.2 From analysis to engineering decisions

The structured analysis should be regarded as a evidence based proposal or as a handoff document for engineering purposes, not as an authoritative specification, as proof of runtime, or as a generated application. Before using it, the engineer should:

1) confirm the proposed purpose and stages against the code, Markdown and known scientific purpose;
2) follow the evidence references and verify the cited support;
3) check the identified inputs, outputs, dependencies and cross-notebook relationships;
4) review exploratory work, manual actions and unresolved questions; and
5) Accept the proposals or revise them or reject them before carrying out the design of the controls, the processing stages, the validation, the feedback and the outputs.

2.3.3 Appropriate uses and limitations

Notebook analysis is especially useful when dealing with long or unfamiliar workflows, in cases where the processing is spread across multiple notebooks, and when dependencies are shown through the use of variables or file handoffs rather than being explicitly documented; it helps to lessen the amount of effort required to find the relevant evidence and gives a framework for discussion with the notebook authors and domain experts.

How useful it is depends on the evidence available. Since workflow intent, clearly defined stage boundaries, exploratory work and potential user-facing inputs all require some kind of contextual interpretation, the results cannot be entirely deterministic. Behaviour that is contained only in external files, in undocumented manual actions, in stored outputs or in the active kernel state may also be outside the scope of an analyser. Although evidence references and deterministic checks make the findings easier to review, human judgement is still needed.

The technique can be put into practice with the use of various models, tools and infrastructure; a practical example of this is the LLM-based analyser of the project.

2.3.4 Practical example: an LLM-based notebook analyser

The reference Analyser demonstrate a practical implementation of using AI to help wit the manual translation of jupyter notebook climate workflows. It employs a large language model to look at the code and Markdown of one or more Jupyter notebooks without running their cells. This enables the analyser to examine the documented workflow and the implemented operations without having to assume that the notebook has been successfully run before.

2.3.4.1 How the notebook analyser works

The analyser can take between one to four notebooks, which are supplied either as uploaded Jupyter notebook files or as GitHub notebook links that are publicly supported. It reads the code and Markdown cells of these notebooks without carrying out the code and then sends the relevant notebooks together to the LLM so that their common context can be taken into account.

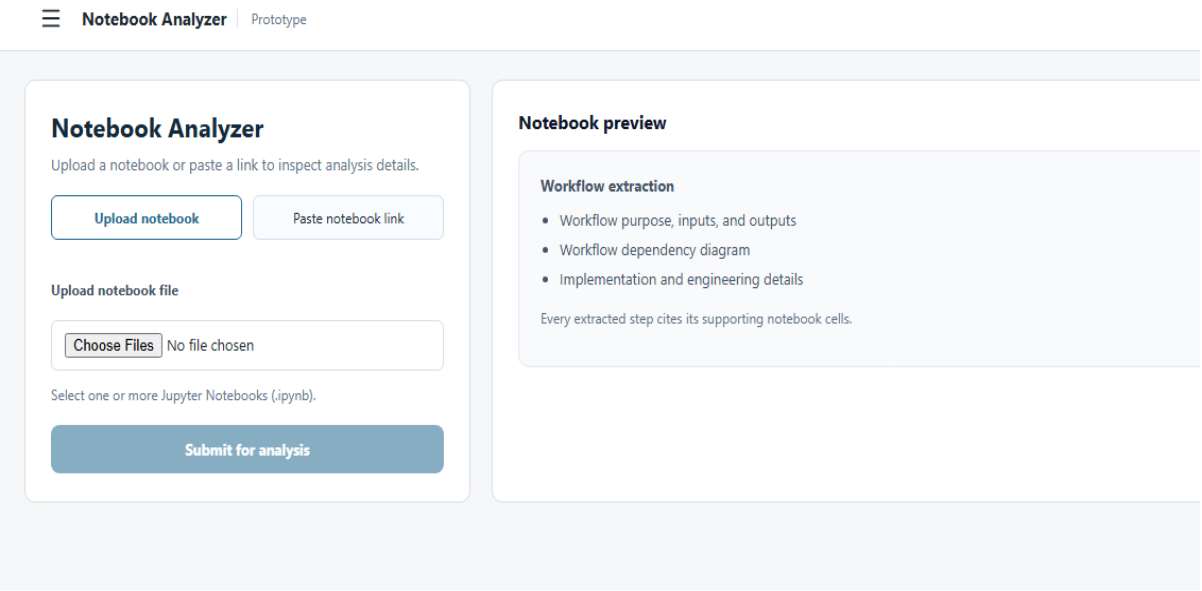


Figure 4 Notebook analyser interface for submitting notebook files or public GitHub links

The response is set out in the form of a structured engineering description which includes details on the purpose of the notebooks, their inputs, configuration, the various stages, the outputs, the dependencies, the exploratory cells, the libraries used, and the unresolved questions. This kind of organisation enables an engineer to examine the main findings and make use of them when planning a manual translation.

2.3.4.2 Evidence and validation

Each proposed workflow stage is linked to supporting notebook cells. For multiple notebooks, proposed relationships must identify both the producing and consuming evidence. Deterministic checks validate the returned structure, step identifiers and dependencies, cited cell indexes, notebook assignments and cross-notebook relationship evidence. These checks improve consistency and traceability, they don't prove any semantic or scientific correctness.

Since the notebook content is analysed without executing it, the analyser would not be able to confirm successful runtime behaviour, generated files, external datasets or values held only in stored outputs or kernel state. Human review, application testing, and scientific validation must therefore remain separate activities.

2.3.4.3 Applying the analyser to the flood notebooks

The notebooks on river flooding hazards and risk, which had been manually translated into the application, were resubmitted to the analyser in order to verify whether the analyser's output matched that of the application in question. This comparison was intended to determine if the analyser was able to reconstruct the main workflow and offer a useful starting point for similar manual translations.

When the hazard and risk notebooks were looked at together, it became clear what the main structure of the workflow was: in the hazard notebook the JRC and Aqueduct flood information was prepared, and in the risk notebook the already-prepared hazard data was used together with land-use information from LUISA and the damage curves. The transition from hazard preparation to the evaluation of economic damage was also noted. Although the exact phrasing and the way the stages of the workflow were grouped differed slightly depending on the model used, the key processing stages and the relationship between the notebooks could still be identified.

The interpretation mostly reflected the main implemented workflow. In both the analysis and the application, the preparation of hazards is separated from the calculation of risks, the high-resolution JRC GeoTIFF being used as input in the risk phase and then combined with land use and damage data to calculate and display the economic impacts. Moreover, the application also displays the Aqueduct products as separate qualitative climate background information. The analyser recreated the workflow shown in the source notebooks, the way in which that workflow was organised, validated, and presented through the application interface being determined by engineering decisions.

This practical example shows how an analyser can give engineers a well-organised basis from which to understand a notebook-based workflow. Yet the resulting description still needs to be checked, since the level of detail and the way in which it is grouped may differ slightly between models. Other practitioner may use different models, prompts, validation rules, infrastructure, or output formats, while retaining the central principles of structured, evidence-linked and reviewable analysis.

Notebook preview

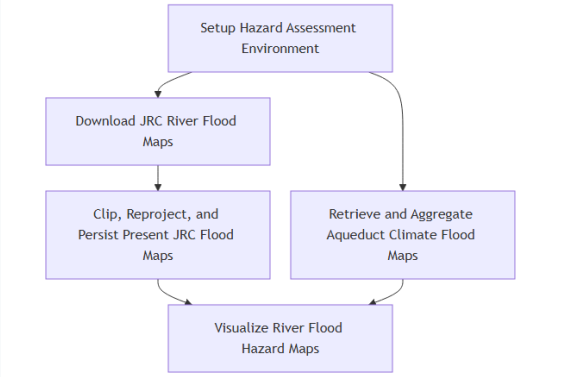
Processed notebook

This two-part workflow evaluates river flooding hazard and economic risk for a selected area of interest. First, high-resolution JRC historical flood maps (10- to 500-year return periods) and global WRI Aqueduct coarse-resolution climate projections (RCP4.5 and RCP8.5 for 2030, 2050, and 2080 across 5 GCMs) are downloaded, clipped to the bounding box, reprojected to EPSG:3035, and persisted as NetCDF and GeoTIFF datasets. Second, the risk assessment loads the preprocessed JRC flood hazard maps, downloads 100m LUISA land cover data, aligns the flood layers to the land cover grid via bilinear resampling, synthesizes composite land-use vulnerability curves from JRC damage functions and custom building ratios, and executes DamageScanner (RasterScanner) to calculate spatial economic loss maps and summary damage tables.

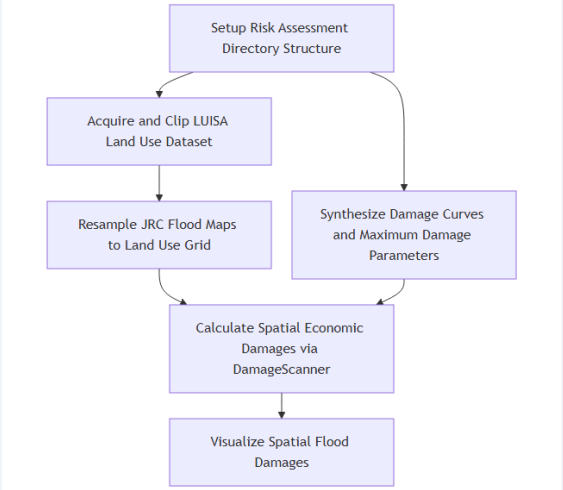
- Name: River Flood Hazard and Risk Assessment Workflow
- Purpose: Assess river flood hazard under historical and future climate scenarios and quantify potential economic damages to land use infrastructure using DamageScanner.
- Inputs: JRC high-resolution river flood maps (Europe_RP{rp}_filled_depth.tif), WRI Aqueduct Floods coarse-resolution river flood maps, LUISA 100m land cover basemap (LUISA_basemap_020321_100m.tif), JRC depth-damage curves (JRC_damage_curves.csv), LUISA damage info template spreadsheet (LUISA_damage_info_curves_template.xlsx)
- Configuration: bbox = [8.480702, 52.959337, 9.128896, 53.216668] (EPSG:4326), areaname = 'Bremen_Germany', return_periods = [10, 50, 100, 200, 500] (JRC flood maps), return_period = 250 (Aqueduct climate scenario analysis), models = ['NorESM1-M', 'GFDL-ESM2M', 'HadGEM2-ES', 'IPSL-CM5A-LR', 'MIROC-ESM-CHEM'], years = [2030, 2050, 2080], scenarios = ['rcp4p5', 'rcp8p5'], landuse_res = 100 (meters)
- Final outputs: FLOOD_RIVER_hazard/plots_Bremen_Germany/Floodmap_overview_JRC_PresentScenario_Bremen_Germany.png, FLOOD_RIVER_hazard/plots_Bremen_Germany/Floodmaps_AqueductFloods_Bremen_Germany_{scenario}.png, FLOOD_RIVER_risk/results_Bremen_Germany/flood_Bremen_Germany_{rp}_damagemap.tif, FLOOD_RIVER_risk/plot_Bremen_Germany/Result_map_Bremen_Germany_damages_overview.png, FLOOD_RIVER_risk/plot_Bremen_Germany/Result_map_Bremen_Germany_rp{rp}.png
- Libraries: contextily, damagescanner, matplotlib, numpy, os, pandas, pooch, rasterio, rioxarray, shutil, xarray

Workflow diagrams

Hazard_assessment_FLOOD_RIVER.ipynb



Risk_assessment_FLOOD_RIVER.ipynb



Notebook relationship

The hazard assessment notebook retrieves, clips, and persists high-resolution present-day JRC river flood maps and coarse-resolution Aqueduct climate scenario flood maps to NetCDF files in the hazard data directory. The risk assessment notebook subsequently loads these NetCDF files to derive spatial bounds, align resolution with land cover data, evaluate climate impact trends, and compute economic damage maps using DamageScanner.

Hazard_assessment_FLOOD_RIVER.ipynb → Risk_assessment_FLOOD_RIVER.ipynb

The hazard assessment workflow exports present-day JRC flood maps as a NetCDF file, which is consumed by the risk assessment workflow to establish bounding spatial extents and align flood layers with land use grids.

Data connection	FLOOD_RIVER_hazard/data_{areaname}/floodmaps_all_JRC_PresentScenario_{areaname}.nc → hazard_data_dir/floodmaps_all_JRC_PresentScenario_{areaname}.nc
Evidence cells	Hazard_assessment_FLOOD_RIVER.ipynb #39, Risk_assessment_FLOOD_RIVER.ipynb #81, Risk_assessment_FLOOD_RIVER.ipynb #87

Hazard_assessment_FLOOD_RIVER.ipynb → Risk_assessment_FLOOD_RIVER.ipynb

The hazard assessment workflow downloads and aggregates Aqueduct climate scenario flood maps into a NetCDF dataset, which is read by the risk assessment workflow to compare future flood depth changes against baseline scenarios.

Data connection	FLOOD_RIVER_hazard/data_{areaname}/aqueduct_floods/inunriver_AllScenarios_AllModels_AllYears_rp00250_{areaname}.nc → data_dir_aqueduct/inunriver_AllScenarios_AllModels_AllYears_rp00250_{areaname}.nc
Evidence cells	Hazard_assessment_FLOOD_RIVER.ipynb #60, Risk_assessment_FLOOD_RIVER.ipynb #92, Risk_assessment_FLOOD_RIVER.ipynb #93

Figure 5 Example analyser output, showing the identified hazard and risk workflow stages

3 Practical considerations and lessons learnt

3.1 Choosing the appropriate approach

The approaches presented offer different ways to make notebook-based climate workflows easier to access, understand and use. Enhancing a notebook with interactive controls and presenting it through Voilà can allow users to explore the workflow while retaining its explanations and visualisations. Manual translation provides scope to reorganise the workflow around user tasks, with dedicated interfaces for selecting inputs, running assessments and interpreting results. AI-assisted analysis can support this translation by helping engineers understand the notebook's processing steps and dependencies.

The starting point is to understand which choices users should be able to make, what guidance they need and how results should be presented. This helps identify where the existing notebook already serves users well and where further adaptation would be useful. Across these approaches, the aim is to make the scientific workflow usable while preserving its meaning and communicating its limitations.

3.2 Preparing the notebooks for interactive use

Enhancing a notebook with Voilà requires more than hiding its code. Since the notebook executes when a session starts, operations such as downloading data, processing rasters or writing files must be separated from interface setup if they should wait for user input. In the implementation described in this guide, these operations are triggered through explicit action buttons.

The interface must also handle actions requested in an unexpected order. Tabs can suggest a sequence, but they do not guarantee that earlier stages have been completed. Each action therefore needs to check its prerequisites and explain what is missing. A useful lesson from the described implementation is that workflow guidance and prerequisite checks work together: navigation shows users what to do, while checks prevent an unavailable intermediate product from being used.

Presentation changes should not unnecessarily repeat scientific processing. The Voilà example separates preparation from plotting, allowing users to adjust display settings without repeating downloads or raster operations. When adapting this approach, practitioners should identify which interactions require recalculation and which only change how an existing result is displayed.

3.3 Making application requirements explicit

Before an application is implemented, practitioners should plan which value are user controllable and how operations should be grouped and which intermediate results must be retained. Where frontend and backend components are separate, their input requirements and handling of workflow state must remain consistent.

The assumptions that the notebook author makes should be turned into explicit application behaviour. Although a local path or a variable set up in a previous cell might be adequate when exploring, the application has to determine where the necessary information comes from and verify that it is available before using it; in the dedicated flood application, for example, the risk stage is given a path to a saved hazard file and the backend then checks whether the file's location is valid and that it exists before carrying on.

Manual tasks must also be taken into account; although the dedicated application allows an area to be selected via the interface, it does not generate the necessary economic information and therefore a pre-prepared area-specific economic workbook is required. It is essential that these prerequisites be made clearly known, specifying what the users have to prepare and when this preparation is needed.

3.4 Reviewing AI-assisted interpretations

When reviewing the analyser's response, attention should be given to the findings that have an impact on implementation, such as the main inputs and outputs, the order in which processing takes place, the manual steps involved and the relationships between the notebooks. Each finding that has consequences should be verified against the referenced cells. In the case of a proposed handoff, both the cell that is producing the data and the cell that is receiving it must support the connection.

Even if a clear description or a convincing flowchart is provided, this does not prove that the response is correct. Although the analyser's deterministic tests can detect structural and attribution issues, it is still possible for a response to pass these tests and yet to misinterpret an operation or fail to take into account an important requirement. It is therefore necessary to carry out separate investigations into missing external information and unresolved questions.

In the informal manual comparison the wording of the responses and the way the stages were grouped were different. Instead of expecting the same responses or a set number of stages, practitioners should decide for themselves whether the interpretation is supported and adequate for the purpose of planning. The interpretation that has been reviewed can then be used to guide the design of the application, while the implementation decisions and validation will still be the engineer's responsibility.

3.5 Checking data and outputs

Checking that an application can find and open a file is only part of validation. Practitioners also need to confirm that the data are suitable for the calculation, including their units, spatial alignment and relevant settings. The role of each dataset should remain clear throughout the translation. In the flood workflow, the original notebooks already use JRC hazard data for economic assessment and Aqueduct products for qualitative climate context. The application preserves this distinction and should make it clear to users.

To check that the scientific processing has been preserved, selected notebook and application results should be compared using the same inputs and settings. Any differences should be investigated, including those caused by intentional changes to the workflow or data handling. An interface that works as expected does not, by itself, confirm that the underlying calculations are correct.

AI-assisted notebook analysis can help identify what needs to be checked, but it cannot carry out this verification. The reference analyser reads code and Markdown without running the cells, so it cannot confirm that the notebook executes successfully or produces correct results. These checks remain part of implementing and testing the application.

3.6 Preparing the operating environment

A workflow accessed through a browser still needs the software, data and file storage that support its processing. Before making an application available to users, practitioners should check that the required software packages are installed, external datasets can be accessed and the locations for input data and generated files are correctly configured. These requirements also apply to the dedicated flood application and need attention alongside interface development.

For AI-assisted analysis, the amount of content submitted needs to fit within the model's capacity. Even when a batch meets the limits on notebook count and file size, its content may exceed the model's context window—the amount of text the model can handle in one request. The reference analyser sends the submitted notebooks together in a single request and does not automatically split oversized content into smaller analyses. Practitioners should therefore consider both the amount of notebook content being sent and the limits of the chosen model and service.

The analyser sends notebook code and Markdown to the configured LLM service. Before submitting notebooks, practitioners should check for credentials or restricted information, remove anything that should not be shared and make sure the service is suitable for handling the remaining content.

4 Key messages and next steps

4.1 Key messages

- **Organize the application around user tasks:** group related processing into meaningful stages and expose only essential parameters.
- **Give users control:** separate interface setup from user-triggered operations in notebooks, and validate prerequisites when actions are requested.
- **Make dependencies explicit:** define inputs, outputs, and manual steps for each stage, including file handoffs.
- **Use AI analysis to support engineering decisions**, but engineers must still design, implement, and test the application.
- **Verify AI proposals against their evidence:** check interpretations, processing stages, and relationships against source notebooks.
- **Confirm scientific meaning alongside application behaviour:** preserve dataset roles and processing logic, and validate results with identical inputs and settings.

4.2 What can you do next?

Take a notebook and make one task usable for someone unfamiliar with its code (e.g., selecting an area to generate a map).

- **Create a small interactive version:** replace a parameter with a widget, tie processing to an explicit action, and test it via Voilà without opening the editor.
- **Sketch a dedicated application:** outline how the task would work with separate interface and processing components, identifying reusable operations and required redesigns.
- **Compare your understanding with the analyser's output:** note the notebook's main stages, inputs, and outputs, then review the analysis for discrepancies (e.g., overlooked steps or unsupported relationships).
- **Test an incomplete workflow:** omit a required input or request a stage before its prerequisite. Ensure the interface clarifies missing steps and guides the user.
- **Have another person test it independently:** observe hesitations or confusion, refine interactions, and retest with the same inputs to confirm consistency.

For multi-notebook workflows, extend the prototype across one handoff: make the first stage's output an explicit input for the next, and submit both notebooks to the analyser to review the connection.

4.3 Further resources

Following resources are directly related to this guideline and provide more in-depth information on main topics discussed:

- Jupyter Notebook documentation: https://jupyter-notebook.readthedocs.io/
- CLIMAAX FLOODS workflow: https://github.com/CLIMAAX/FLOODS
- Voila Framework documentation: https://voila.readthedocs.io/en/stable/

Table 3 shows other recommended platforms for further reading and access to related ClimEmpower resources.

Table 3: Recommended Platforms

Source platform	Short definition	Relevant materials
ClimEmpower Project Results — climempower.eu/results/	Official ClimEmpower platform for project results and outputs	Project deliverables; resilience recommendations; risk-map information; project tools and software; scenarios; data and knowledge assessments; stakeholder engagement results
ClimEmpower Educational Library — climempower.eu/knowledge-hubs/	ClimEmpower collection of educational materials and selected knowledge resources	Educational materials; Guides and manuals; presentations; learning resources; climate-hazard and sector resources; target-group resources; external knowledge resources
CORDIS — cordis.europa.eu	European Commission platform for EU-funded research and innovation	Project descriptions; public deliverables and reports; publications; project partners; project results and outputs
Climate-ADAPT — climate-adapt.eea.europa.eu	European platform for climate change adaptation knowledge and practice	Adaptation strategies and policies; case studies; climate risks and impacts; adaptation options; tools; indicators; research and innovation projects
ClimEmpower Zenodo Community — zenodo.org/communities/climempower	Open repository for ClimEmpower research outputs and project materials	Datasets; journal publications; conference papers and posters; educational materials; other project outputs