DATA CURATION NETWORK

Jupyter Notebooks: A Primer for Data Curators

Topic	Description
File Extension	.ipynb
MIME Type	https://jupyter.readthedocs.io/en/latest/reference/mimetype.html
Structure	Browser-rendered composite digital asset: Notebook file (.ipynb); Notebook app; kernel
Versions	4.0.0 - 5.7.0 (previously IPython Notebook)
Primary fields; areas of use	Not discipline-specific; can be used by anyone who writes code in a language with a supported kernel
Source and affiliation	Project Jupyter
Metadata standards	Codemeta; CFF; Jisc/SSI Guidance; discipline-specific keywords
Tools for curation	nbconvert, nbviewer, repo2docker; CodeMeta crosswalks; CodeMeta tools; CFF tools
Date created	February 1, 2019
Created by	Daina Bouquin; Sophie Hou; Matthew Benzing; Lee Wilson
Updated	Date, GitHub link w/ release notes

Suggested Citation: Daina Bouquin; Sophie Hou; Matthew Benzing; Lee Wilson. (2019). Jupyter Notebooks: A Primer for Data Curators. Retrieved from the University of Minnesota Digital Conservancy. http://hdl.handle.net/11299/202810.

This work was created as part of the Data Curation Network "Specialized Data Curation" Workshop #1 co-located with the Digital Library Federation (DLF) Forum 2018 in Las Vegas, Nevada on October 17-18, 2018. See also: Primers authored by the workshop attendees at DLF. http://datacurationnetwork.org.

Introduction

Jupyter Notebooks are composite digital objects used to develop, share, view, and execute interspersed, interlinked, and interactive documentation, equations, visualizations, and code. Researchers seeking to deposit software, in this case Jupyter Notebooks, in repositories do so with the expectation that repositories will provide documentation explaining "what you can deposit, the supported file formats for deposits, what metadata you may need to provide, how to provide this metadata and what happens after you make your deposit" (Jackson, 2018a). This expectation is not necessarily met by repositories that currently accept software deposits and complex objects like Jupyter Notebooks. This guide is meant to both inform curatorial practices around Jupyter Notebooks, and support the development of resources that meet researchers' expectations to ensure long-term availability of software in curated archival repositories. Guidance provided by Jisc[1] and the Software Sustainability Institute[2] outlines three different kinds of software deposits: a minimal deposit, a runnable deposit, and a comprehensive deposit (Jackson, 2018b). This primer follows this same conceptual framework in dealing with Jupyter Notebooks, which even in their static, non-executable form, can be used to document how scientific research was carried out or be used as teaching models among many other use cases.

Jupyter Notebook Format Description

A Jupyter Notebook is a file used in conjunction with a suite of tools that allow users to create and share documents that contain runnable code, equations, data visualizations, and other interactive material. While Python is the most common language associated with Jupyter Notebooks, they can be used with code written in over 40 different programming languages. Jupyter Notebooks' versatility enables them to be used in any number of disciplines and for various purposes, and while they are very popular in the sciences, they are also used in the social sciences and the humanities. Because Jupyter Notebooks are meant to be interactive and constructed using a multitude of programming and spoken languages, they are especially challenging for curators to work with. Any curation and archiving activity needs to be done in such a way as to not inhibit a future user's need to adapt the code contained within the Notebook file. Similarly, when a future user extracts deposited Notebook files, metadata, and supplemental material from the archive, curation and archiving activities should have had no degrading influence on the level of functionality that a depositor enabled with their initial deposit. For example, rather than zipping files on the depositor's behalf, it is preferable for curators to request that depositors pack and unpack their content prior to making their deposit to allow the them to check that files function as intended when unpacked.

To open a Jupyter Notebook file, a curator would need to have installed Python and Jupyter (using either pip or Anaconda[3]) and be familiar with using the Terminal (Mac/Linux), Command Prompt, or Bash (Windows).[4] Once opened, Jupyter Notebooks have a browser-rendered user interface composed of "cells" and clickable buttons to execute tasks. A cell is a multiline text input field where a user can enter and execute code or a markup language called Markdown. Markdown handles text formatting, linking, and the display of images. Behind the Notebook cells is a kernel that runs the processes needed for each cell to function. Code cells often require dependencies and specific input parameters, and may be run in any order, which is both a strength and a weakness.[5]

Once rendered in the user's browser, a Notebook can be exported in the following formats:

- Notebook (.ipynb)

[1] https://www.jisc.ac.uk/
[2] https://www.software.ac.uk/
[3] https://jupyter.org/install
[4] https://jupyter.readthedocs.io/en/latest/running.html#running
[5] https://bit.ly/2Tw2aIo

- Python (.py)
- HTML (.html)
- Markdown (.md)
- reST (.rst)
- PDF via LaTeX (.pdf)

The following are useful tools for working with Jupyter Notebook files and curating metadata associated with them:

- Rendering Notebook files: nbviewer[6]
- Generating PDFs: nbconvert[7]
- Building Docker containers: jupyter-repo2docker[8]
- Generating and converting CodeMeta.json: CodeMeta Tools[9]
- Generating and converting CITATION.cff: CFF Tools[10]

Deposit Requirements

The following elements outline recommendations for repositories accepting Jupyter Notebook submissions. Minimally required files and metadata will support the ability to open and cite the Notebook, but additional functionality should not be expected without requiring additional files and more comprehensive metadata.

File Requirements:

- **Minimally required files:**
  - .ipynb (cells run with results viewable)
  - README (.txt or .md)
  - LICENSE (.txt or .md)
- Additional files to request:
  - PDF of the Jupyter Notebook (export from Jupyter web application or nbviewer)
  - reST export of the Jupyter Notebook (export from Jupyter web application)
  - CodeMeta.json
  - CITATION.cff
  - Sample datasets and documentation (see below)
  - Container metafile (e.g. docker, singularity, reprozip)
    - Can be created using jupyter-repo2docker
    - Can be published separately with execution instructions; link this to the Jupyter Notebook record
  - Release of the full repository of files associated with .ipynb when applicable
    - Recommend minting a software DOI for the code repository (Fenner et al., 2018)
    - Provide guidance on how to mint a software DOI (e.g. assigning a software DOI via Zenodo[11])

Metadata Requirements:

- **Minimal submission:** baseline description; enables user to view and cite the Notebook

[6] https://github.com/jupyter/nbviewer
[7] https://github.com/jupyter/nbconvert
[8] https://github.com/jupyter/nbconvert
[9] https://codemeta.github.io/tools/
[10] https://citation-file-format.github.io/#/tools
[11] https://guides.github.com/activities/citable-code/

    - Jupyter Notebook title
    - Author(s)
    - Jupyter implementation details
        - Jupyter version
        - Distribution (e.g. Anaconda)
        - Kernel version
    - README
        - Documents what the Jupyter Notebook is for
        - Request that this file include citation(s) to third-party algorithms and analyses
        - Recommend code comments within the Notebook file itself in addition to the README file
    - License information
    - Alternate identifiers and supplemental links associated with the Notebook
- **Runnable submission:** allows another researcher to execute the Notebook locally using sample data and files provided by the depositor[12]; minimal submission metadata plus:
    - User documentation
        - Instructions to support configuration needed to execute the Notebook and code cells
        - Sample input and output files
    - CodeMeta.json
        - Document required software dependencies
        - Recommend additional machine actionable dependency documentation (e.g. requirements.txt)
    - CITATION.cff for the Notebook
        - Preferred citation; should enable native software citation
- **Comprehensive metadata:** minimal and "runnable" requirements plus:
    - Developer documentation
        - Include test code and description of expected results
    - Narrative description of how the code implemented in the Notebook works and what it does
    - Documentation about the computing ecosystem (e.g. CodeMeta.json: targetProduct, processorRequirements)

Key Curatorial Questions

Once a decision has been made to accept and curate Jupyter Notebook submissions in an archival repository, the following questions should be considered with each submission:

1. What are the depositor's expectations for the Notebook's future functionality once the deposited files are exported from the archival repository?
2. Does the submission include minimally required files and metadata to enable the expected functionality?
3. Is the Notebook self-contained?
4. Is the Notebook a standalone object or one of many products resulting from a project?
    a. Examples:
        i. Notebook that is a stand alone object: USGS Python for Data Management[13]
        ii. Notebooks that supplement other digital objects: Starry[14]

[12] This assumes the Notebook is self-contained. How to best archive Notebooks that are not self-contained is an unresolved issue.

[13] https://bit.ly/2sBF3jH

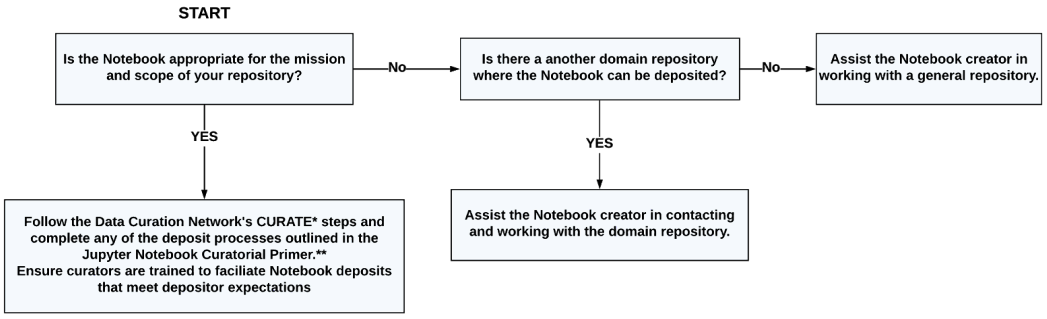
[14] https://arxiv.org/abs/1810.06559

   b. Were supplemental files deposited along with the Notebook?
      i. Is information about supplemental files included within the Notebook or in separate files?
      ii. If separate files, can those files be opened and read?
   c. Are there multiple Notebooks in the deposit?
      i. If multiple Notebooks were deposited together, do they require different metadata to meet the depositor's functionality expectations?
5. What are the technical characteristics of the Notebook? Including:
   a. File size
   b. Availability of alternate format(s)
   c. Availability of additional copies
6. Who is the intended user community?
7. Are there any specific search, discovery, and/or access needs?
8. Are there any specific usage metrics requirements?
9. Is the Notebook expected to be replaced or updated by a newer version at a later date?
10. Is the Notebook peer-reviewed?
11. Are there any confidentiality/ethics concerns associated with the Notebook?

Decision Trees (view online)

The following decision trees[15] illustrate questions and actions that should be considered when determining whether or not to accept a Jupyter Notebook submission into a particular repository, as well key questions curators should consider when evaluating Jupyter Notebook submissions.

Repository Suitability

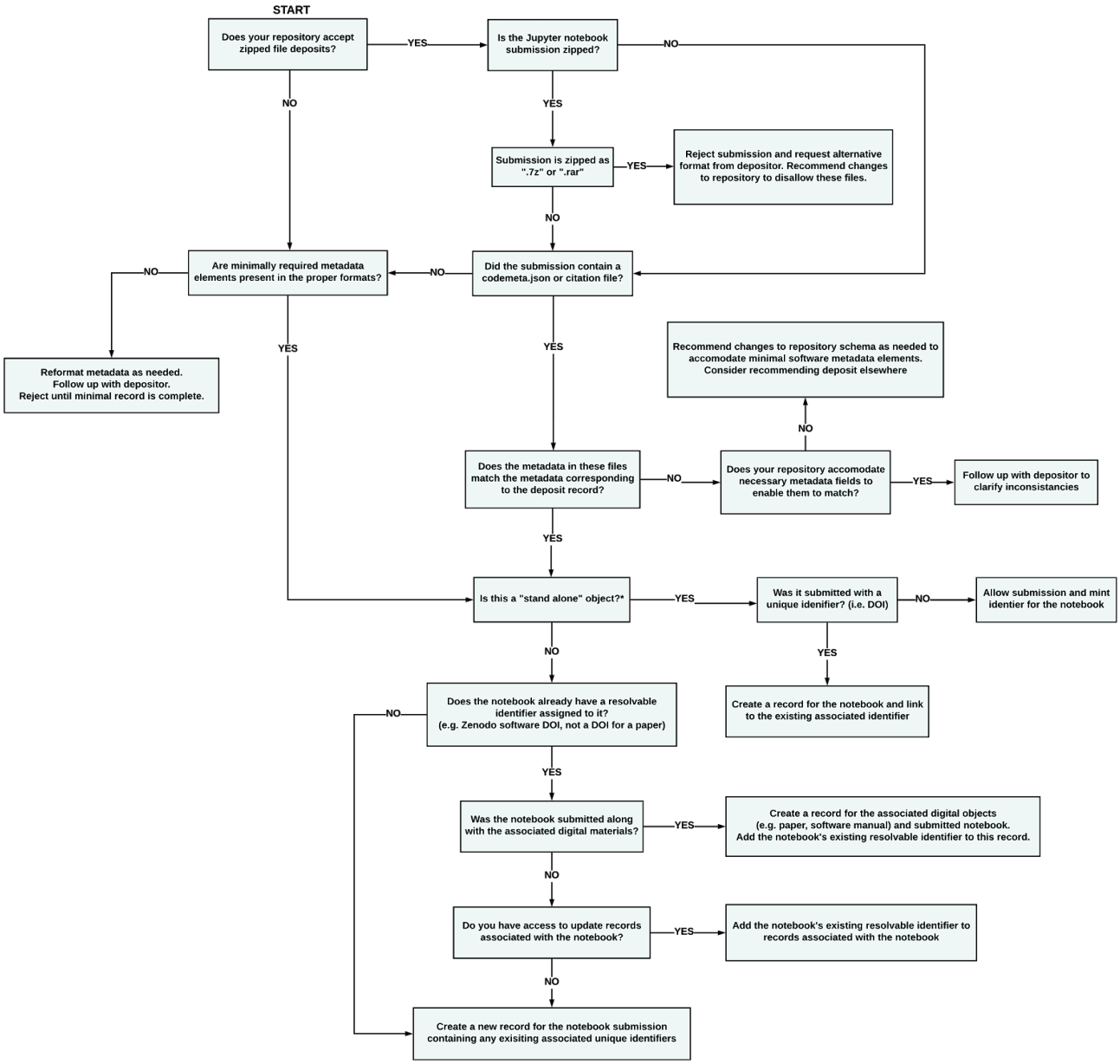


* https://datacurationnetwork.org/home/resources/

** [Link to published primer]

[15] https://www.lucidchart.com/documents/view/4848c483-1267-499c-9172-3a2782abfaaf/0

Curatorial Activities



* Repositories should request alternate identifiers and supplemental links when accepting Jupyter notebook deposits

Additional Recommended Reading

- Software Deposit Guidance for Researchers
  - https://zenodo.org/record/1327310
- Ten Simple Rules for Reproducible Research in Jupyter Notebooks
  - https://arxiv.org/abs/1810.08055
- How IPython and Jupyter Notebook work
  - https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html
- Developing maintainable software
  - https://www.software.ac.uk/resources/guides/developing-maintainable-software

- Does it make sense to apply the FAIR Data Principles to Software?
    - https://indico.cern.ch/event/588219/contributions/2384979/attachments/1426152/2189855/FAIR_Software_Principles_CERN_March_2017.pdf
- FAIR is not fair enough
    - https://danielskatzblog.wordpress.com/2017/06/22/fair-is-not-fair-enough/
- Compact identifiers for software: The last missing link in user-oriented software citation?
    - https://danielskatzblog.wordpress.com/2018/02/06/compact-identifiers-for-software-the-last-missing-link-in-user-oriented-software-citation/
- nbconvert documentation
    - https://nbconvert.readthedocs.io/en/latest/
    - Really any of the documentation about Project Jupyter
- Reproducible Research using Jupyter Notebooks Course
    - https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/
- Jupyter Notebooks and reproducible data science
    - https://markwoodbridge.com/2017/03/05/jupyter-reproducible-science.html
- I don't Like Notebooks, Joel Grus
    - https://bit.ly/2Tw2aIo

References

Fenner, M., Katz, D. S., Nielsen, L. H., & Smith, A. (2018, May 17). DOI Registrations for Software. *DataCite Blog*. doi: https://doi.org/10.5438/1nmy-9902

Jackson, M. (2018a). Software Deposit: How to deposit software (Version 1.0). *Zenodo*. http://doi.org/10.5281/zenodo.1327327

Jackson, M. (2018b). Software Deposit: What to deposit (Version 1.0). *Zenodo*. http://doi.org/10.5281/zenodo.1327325