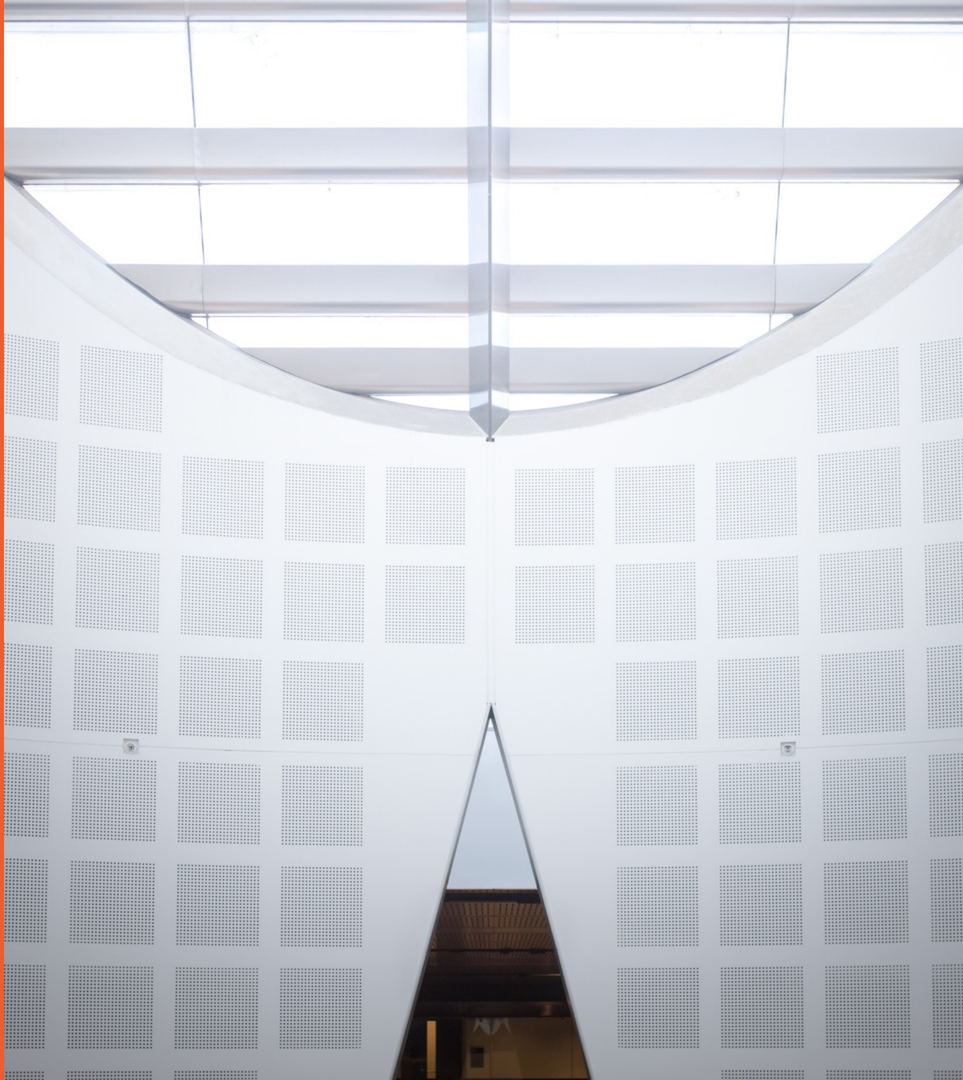
Achieving reproducibility with computational notebooks

Tanaka Chitete
Student, Bachelor of Advanced Computing (Honours)

Dr. Rahul Gopinath
Lecturer, School of Computer Science

THE UNIVERSITY OF SYDNEY



Context

Reproducibility for notebooks differs from scripts

Computational notebooks have a broader range of artifacts:

1. Source code
2. Data and parameters used to run the code
3. Text output from the code
4. Visualisations
5. Tables

Research questions

Achieving reproducibility with notebooks

Addressing the following research questions will achieve this end:

1. How could we achieve the saving and restoration of a partially-computed notebook?

2. How could we achieve the lightweight sharing of a notebook with minimal expectations on the receiver?

3. How could we achieve the collaboration of multiple users on a single notebook?

Critical literature review

Replicating partially-computed notebooks

Reviewed works are either:

1. Snapshot-based
   - Wannipurage et. al., 2022
   - Jurič et. al., 2021
2. Provenance-based
   - Pimentel et. al., 2015
   - Pimentel et. al., 2017

Critical literature review

Resolving dependencies

Existing research addresses this problem through:

1. Static dependency resolution
    - Wang et. al., 2021
2. Dynamic dependency resolution
    - Zhu et. al., 2021

Critical literature review

Supporting collaboration

Prevailing literature implements:

1. Version control
   - Kery et. al., 2018
2. Synchronous editing
   - Wang et. al., 2019

Research gap

No singular work achieves state replication, dependency resolution, *and* facilitation of collaboration for general purpose computational notebooks.

Proposed solution

A fully self-contained JupyterLite environment implemented within an HTML file.

Research methods

1. Encapsulate an empty JupyterLite environment
    1. Embed local files into single HTML file
    2. Embed remote files into same HTML file
2. Instantiate a JupyterLite environment given user input
    1. Embed input notebook into environment
    2. Install dependencies from input requirements file within environment

Research plan

	Jun	Jul	Aug	Sep	Oct	Nov
Implementation						
Development	🟧	🟧	🟧	🟧		
Evaluation				🟧	🟧	

Research plan

	Jun	Jul	Aug	Sep	Oct	Nov
Thesis						
Literature review	🟧	🟧	🟧	🟧	🟧	
Introduction	🟧	🟧				
Background		🟧	🟧			
Methodology			🟧	🟧		
Evaluation				🟧	🟧	
Discussion					🟧	🟧
Conclusion						🟧

Bibliography

1. D. Wannipurage, S. Marru and M. Pierce, “A framework to capture and reproduce the absolute state of Jupyter Notebooks”, *Proceedings of the 2022 Practice and Experience in Advance Research Computing conference*, 2022.
2. M. Jurič, S. Stetzler and C. T. Slater, “Checkpoint, restore, and live migration for science platforms”, *Proceedings of the 2021 Astronomical Data Analysis Software and Systems conference*, 2021.
3. J. F. Pimentel, L. Murta, V. Braganholo and J. Freire, “noWorkflow: a tool for collecting, analyzing, and managing provenance from Python scripts”, *Proceedings of the 2017 Very Large Database endowment,* 2017.
4. J. F. Pimentel, L. Murta, V. Braganholo and J. Freire, “Collecting and analyzing provenance on interactive notebooks: when IPython meets noWorkflow”, *Proceedings of the 2015 Theory and Practice of Provenance conference*, 2015.
5. J. Wang, L. Li and A. Zeller, “Restoring execution environments of Jupyter Notebooks”, *Proceedings of 2021 International Conference on Software Engineering”*, 2021.
6. C. Zhu, R. K. Saha, M. R. Prasad and S. Khurshid, “Restoring the executability of Jupyter Notebooks by automatic upgrade of deprecated APIs”, *Proceedings of 2021 International Conference on Automated Software Engineering*, 2021.
7. M. B. Kery and B. A. Myers, “Interactions for untangling messy history in a computational notebook”, *Proceedings of the 2018 Visual Languages and Human-Centric Computing symposium*, 2018.
8. A. Y. Wang, A. Mittal, C. Brooks and S. Oney, “How data scientists use computational notebooks for real-time collaboration.”, *Proceedings of the 2019 Human-Computer Interaction conference*, 2019.