Experience with Reproducibility and Consistency in Writing an Academic Paper

Joseph Wonsil[1], Nichole Boufford[2], Margo Seltzer[1]

[1]The University of British Columbia [2]Oracle Labs, work performed at UBC

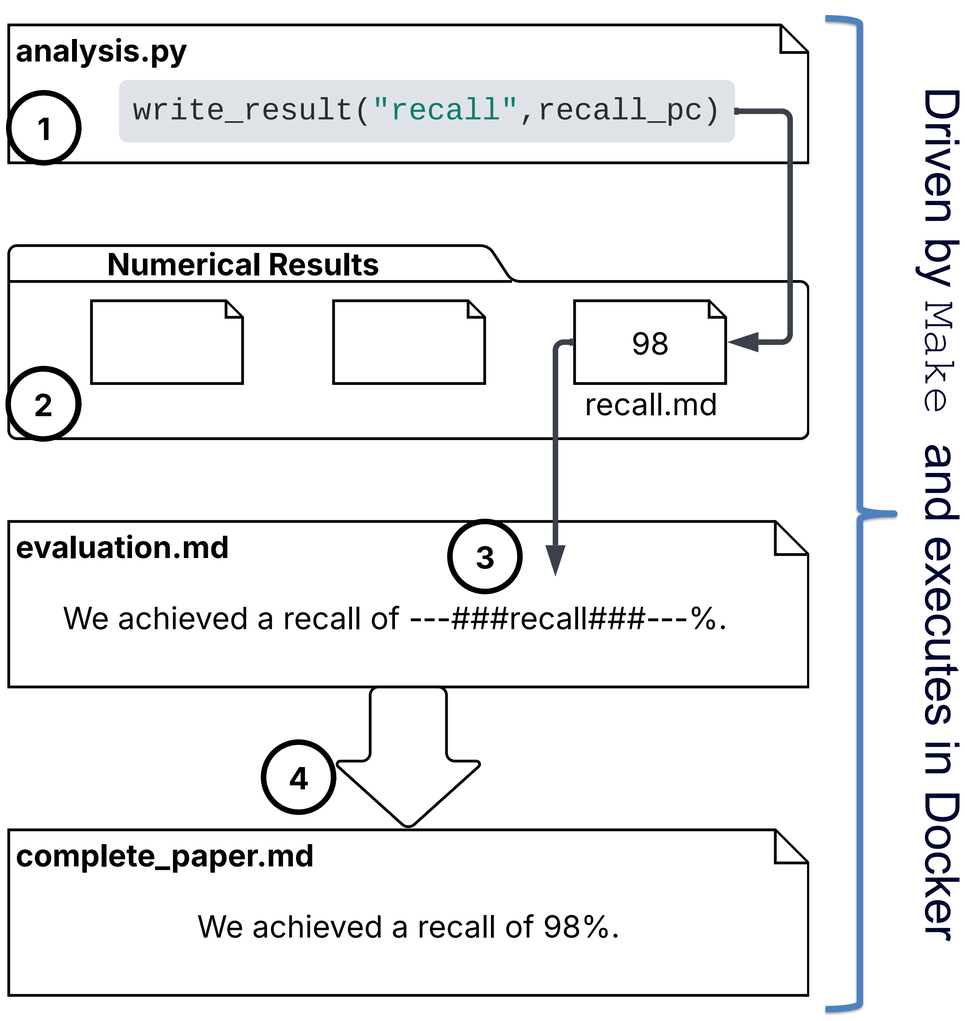
Introduction

The iterative and synchronous processes of writing code for an analysis and writing a paper based on that analysis can lead to inconsistencies between the data, the figures in the paper, and the prose. We wrote a consistent and reproducible paper driven by standard software engineering tools and discuss our lessons learned.

Requirements

1. Each build of the paper should reflect any changes made to the code and results
2. Referencing a result while writing the paper should be distinguishable from the rest of the prose.
3. The pipeline should be able to execute on others' machines with minimal effort.

Implementation



Workflow to ensure analysis-to-paper connection

Results are saved as macros in directories (1)(2), referenced in the prose via their macro (3), and the values inserted by our code (4) before compilation into its final form

```
numerical_results := $(wildcard numerical_reuslts/*.md)

figures := $(wildcard figures/*.png)

analysis.py: analysis.ipynb
    jupyter nbconvert analysis.py --to python && python analysis.py

complete_paper.md: macro_sub.py analysis.py $(numerical_results) $(figures) intro.md background.md methods.md evaluation.md conclusion.md
    python macro_sub.py

main.docx: complete_paper.md citations.bib
    pandoc complete_paper.md \
    -f markdown+smart \
    -- bibliography citations.bib \
    -o main.docx
```

Representative Makefile

Make drives the whole process, ensuring when the paper is compiled any new results are inserted into the document

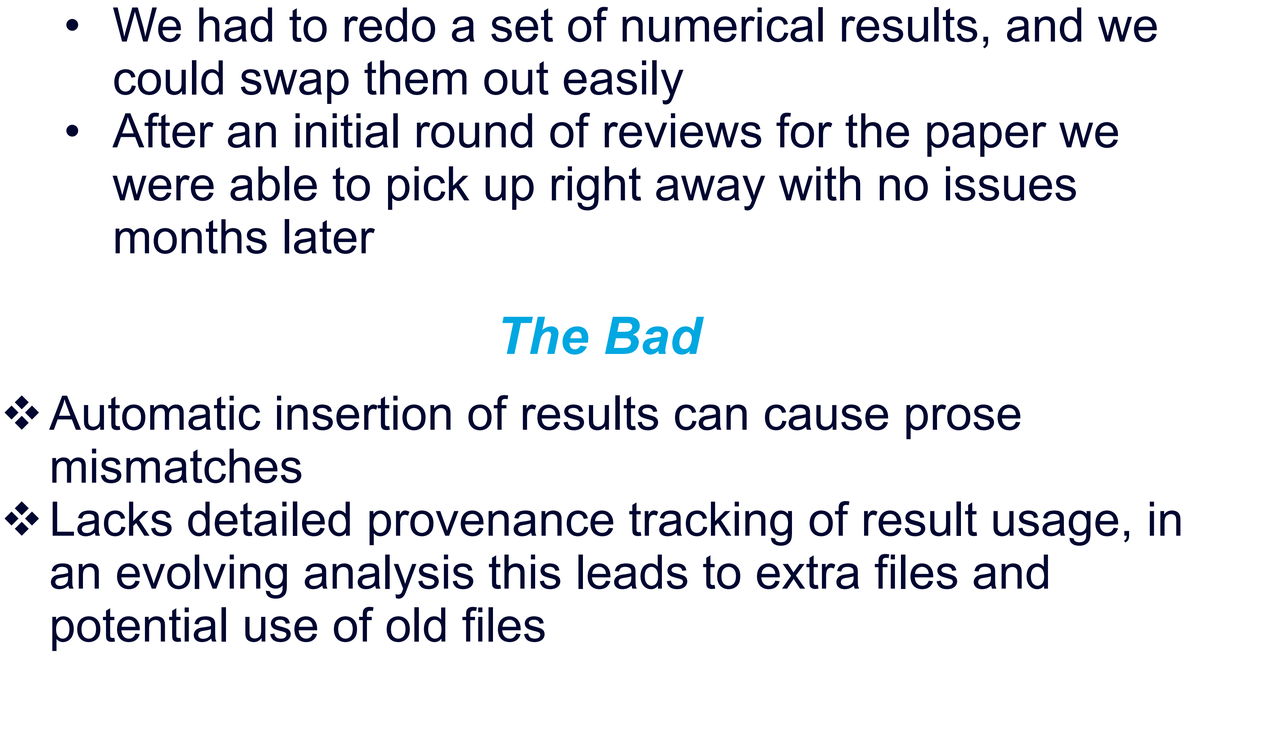
Lessons Learned

The Good

- Achieved our requirements
  - `Make` ensures new code always executes and new results appear in the paper
  - Macros provide searchable patterns to find result references
  - Docker provides a portal environment used for both development/writing and reproducibility
- Effort paid off multiple times:
  - The use of familiar tools ensured adoption was easy and seamless
  - We had to redo a set of numerical results, and we could swap them out easily
  - After an initial round of reviews for the paper we were able to pick up right away with no issues months later

The Bad

- Automatic insertion of results can cause prose mismatches
- Lacks detailed provenance tracking of result usage, in an evolving analysis this leads to extra files and potential use of old files



Examples of file build-up over time

A script might have generated a file at one time (1), might continue to generate it but it's not currently needed (2), or might have generated it at one time, does not anymore, but the file is still being used (3)