Panmirror Planning

J.J. Allaire

May 2020

## Overview

The Panmirror project has several goals, including the creation of a new set of tools for scientific writing and communication, improving the RStudio toolchain, and making a contribution to the Python data science ecosystem.

These goals and the tactics required to achieve them have significant overlap. This document is an attempt to clearly enumerate all of the goals (and their implied tactics) with an eye towards prioritizing short and intermediate term work.

## Improve scientific writing and communication

### Goals

1. Evolve the standard scientific publishing toolchain from print-first to multi-medium.
   * Interoperate with existing print-dominated publishing ecosystem.
   * Leverage the strengths of the web as a connected, interactive platform for communications.
2. Make reproducible research and publications the norm rather than the exception.
3. Facilitate fluid, open collaboration around scientific and technical manuscripts.
4. Reduce the cost of peer review and publication, so that [Platinum Open Access](https://ask-open-science.org/1013/what-is-platinum-open-access) becomes the norm.
5. Provide an open-source toolchain for scientific communication (broad accessibility and guarantee that it won't be hijacked by a narrow commercial interest).

### Strategy

1. Replace LaTeX and MS Word as the preferred toolchain(s) for technical writing:
   * Technical writing capabilities of LaTeX
   * Ease of use and collaborative features of MS Word
2. Provide a best-in-class (by a wide margin) scientific writing tool.
3. Make computational workflow/assets a fundamental part of writing and publishing.
4. Provide flexible collaboration tools:
   * Google Docs style editing experience for technical manuscripts.
   * API for custom review and publication workflows
   * Global account system for cross-institution collaboration (a la GitHub)
5. All of the above as the basis for an extensible platform upon which the goals above can be achieved (by a combination of us and third parties)

### Tactics

1. Create a pandoc wrapper (panmirror command line tools) that provide:
   * Custom formats
   * Build/project system
   * Flexible incorporation of computations
     + Embedded (like Jupyter/Rmd)
     + External (in parallel w/ document)
   * Package management (including LaTeX packages)
2. Create a visual editor for Pandoc
   * Deliver as VS Code add-in (early adopters)
   * Deliver as a standalone tool (via Theia) which is also a front-end for panmirror command line tools.
3. Collaborative visual editing of Pandoc markdown
   * Online service (similar to Overleaf)
   * On premise version (open source and pro)
   * Interop w/ desktop client

## Improve the RStudio toolchain

### Goals

1. More accessible and easy to use R Markdown editing & computation experience.
2. Facilitate collaboration with technical and non-technical users on R Markdown documents.

### Tactics

1. Create a new R Markdown/Notebook user experience based on panmirror
2. Collaboration server for R Markdown docs
   * Public service
   * RStudio Team

## Improve the PyData Toolchain

### Goals

1. Make a distinctive contribution to Python data science tools
2. Have a more direct relationship with Python data science users

### Tactics

1. Create a new Jupyter front-end based on panmirror (and related collaborative tools)
   * Initially deployed as a VSCode add-in
   * Subsequently deployed as part of a new Theia-based IDE
2. Python oriented extensions to panmirror command line tools (e.g. rendering .ipynb, using .ipynb or .py as a source of reproducible computational artifacts). Effectively "R Markdown for Python".
3. Collaborative service / server for new Jupyter front-end and "R Markdown" equivalent.

## Project Planning

This table summarizes the various tactics discussed above along with which group of users they are compelling to. This in turn should help us prioritize short and intermediate term work.

### Roadmap

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R/RStudio | Python | Author | Collaborator |
| **Visual Editor** |  |  |  |  |
| RStudio embedding | • |  |  |  |
| VS code add-in |  | • |  |  |
| Theia-based technical writing IDE |  |  | • |  |
| **Command Line Tools** |  |  |  |  |
| Pandoc project / packaging custom formats |  |  | • |  |
| R extensions for command line tools | • |  |  |  |
| Python extensions for command line tools |  | • |  |  |
| **Notebook** |  |  |  |  |
| RStudio embedding | • |  |  |  |
| VS code add-in (Jupyter front-end) |  | • |  |  |
| Theia-based data science IDE |  | • |  |  |
| **Collaboration server** |  |  |  |  |
| Public hosted service | • | • | • | • |
| On-Prem / VPC server |  |  | • | • |
| Integrate into RStudio Team | • | • |  | • |

Note that the **Collaborator** user category is not one that we will explicitly target with product offerings but are a critical constituency nonetheless. They will typically make small changes to text (not code) and ask questions about the content. They’ll generally already be familiar with track changes / commenting in traditional editors, and want to use these with technical content.

### Work Underway

There are two committed work items (both for RStudio v1.4) from the roadmap above:

* Visual editor embedded into RStudio (v1.4)
* R Markdown/Notebook UX for visual editor

We expect that this will provide a more pleasing and productive R Markdown experience for existing users, a smoother on-ramp into R Markdown for new users, and new modality for delivering educational experiences where we combine fully rendered content and computations (with the added benefit over learnr that users can easily take away their work w/ them).

### Future Work

For the purposes of planning future work, we’ll look at how we can deliver various subsets of the roadmap that are compelling to the 3 identified user groups.

One important distinction for this discussion is between supporting documents with *embedded* computations (Notebooks) and documents with *external* computations (incorporating existing computations into a long form document). Embedded computations are generally a feature of interactive data analysis tools, whereas external computations are generally a feature of writing tools (e.g. embedding a spreadsheet table inside a Word or Google doc). It’s important to recognize that even though they sit on the same technical foundation these are different use cases with different optimal delivery vehicles (data analysis tool vs. writing tool).

#### Technical Authors

Technical authors consist of users of LaTeX editors (LyX, Overleaf, etc.) as well as users of scholarly markdown derivatives like Pandoc. They also consist of users currently using MS Word for writing that would be better handled by a text-based format (but they are have too much affinity for Word’s usabiilty and/or collaborative capabilities.

It’s also important to note that the “whole product” for technical authors doesn’t necessarily involve *embedded* computations (notebooks) but rather looks more like a traditional technical writing tool (LyX, Overleaf, Authorea, etc.) that supports *external* computations.

Here are the individual features along with several coherent subsets of users that grouping of them could address:

|  |  |  |  |
| --- | --- | --- | --- |
| Features | Early Adopters | Academic Market | Entire Market |
| Visual editor (VS code add-in) | • | • | • |
| Command line tools | • | • | • |
| Collaboration service (public) |  | • | • |
| Collaboration server (on-prem) |  |  | • |
| Standalone writing tool |  | • | • |

Here we want to consider what is the set of features to target for an initial release. Note that if we wanted to go extremely minimal we could even release any one of the visual editor, command line tools, or collaboration service to the existing early adopters (all have their individual utility).

I think that going after the entire market for the first release is unnecessary, so that leaves two paths:

1. Target early adopters with core command line tools and a VS code add-in for the visual editor. This would provide Pandoc users with roughly the equivalent capabilities of RStudio / R Markdown. If support for Python computations was included in the command line tools it could also get a lot of adoption among Python users looking for better tools for reproducible documents / research.
2. Target the academic market more broadly by creating a full standalone writing tool and collaboration service (thus clearly positioning panmirror as a next generation LaTeX that can fully replace the LaTeX / LyX / Overleaf toolchain).

The first path is a full subset of the second path so could also be viewed as an intermediate ship vehicle. It also has the benefit of not requiring deliver of a full end-to-end writing application (at the cost of only being attractive to users willing to install/manage VS Code).

This work will be undertaken by the Panmirror team — JJ, Charles, and (for the time being) Joe.

#### RStudio Users

As described above we’ll be delivering a major upgrade to R Markdown / Notebooks in RStudio IDE v1.4.

The next logical iteration after that is to deliver both public and on-premise collaboration capabilities (that are also well integrated w/ the desktop client). This will provide not just collaborative editing of prose, but fully collaborative data analysis. These features we anticipate building as part of RStudio IDE v1.5.

Another set of capabilities of interest to RStudio users will be the panmirror command-line tools. This will represent a next-generation “cross-language” version of R Markdown, which will supported *embedded* and *external* computations equally well. It also may represent an opportunity to revisit things like chunk caching with a more sophisticated approach.

This work will be undertaken by the IDE team (in collaboration with the Panmirror team).

#### Python Users

As our professional products move to embrace Python users, we have a couple of related goals:

1. Begin making meaningful contributions to the PyData ecosystem (we are doing some of this now w/ the Arrow project).
2. Creating a footprint in front-end tooling.

For #2, our current approach is to rely on existing tools (currently JupyterLab and VS Code). However, these tools face some risk of underdevelopment (JupyterLab) or development in a direction highly slanted to a vendor (VS Code). Further, there are nascent efforts to build proprietary front-ends to Jupyter (DeepNote, PyCharm) which could capture enough mind share that users don’t actually prefer the Python tools in RSP to these proprietary offerings.

Panmirror presents two parallel opportunities to contribute to PyData:

1. Create an "R Markdown for Python" by adding comprehensive PyData features to panmirror command line tools. This work is essentially already committed to as reproducible computational documents are a core goal of panmirror command line tools.
2. Create a next generation Jupyter front-end based on Panmirror and the work we do on R Notebooks in the IDE.

The idea of a next-generation Jupyter front-end is opportunistic and can be dialed up or back as necessary. You can imagine this work along a continuum:

* Create a new Juypter front-end as an extension to VS Code
* Create a new Jupyter front-end as an extension to JupyterLab
* Create a new standalone data science IDE (based on Theia) that includes the notebook and other PyData tools.
* Progressively migrate functionality from the RStudio IDE into the new IDE, until some point in the future when we have one cross-language data science IDE that we actively develop.

None of the Python work described above is currently planned or resourced. If we collectively believe that the work is high-priority then we should probably add some resources to ensure we can deliver it in a timely fashion.