Distill.pub

An HTML interactive journal

Quentin Bammey

April 16, 2021

Centre Borelli
ENS Paris-Saclay

What is Distill?

What is Distill?

- An online peer-reviewed research journal
- Articles are web pages in HTML+JS
- Where IPOL focuses on code reproducibility, Distill focuses in **communication and visualization** of results.
- Most prominent feature: interactive diagrams in the articles
- Either novel research, or extensive analysis of existing works and problems

Why use Distill?

- Much more flexible than traditionnal PDF journals
- Especially important for ML where interactivity is often needed for good visualization of data and results.
- Can be (but isn't often) coupled with a full integrated demo where the user inputs its own data and see the results
- No size limit: can be as short or as long as it needs to be
- Once the interface is done, allows easy experiments: it as value not only as a publication, but also as an analysis tool to dissect successes and failures of algorithms.

Why not use Distill?

Distill also has many drawbacks

- The template they offer is very bare, akin to a journal LaTeX template.
- In other words: Distill allows you to create interactive diagrams, but you must do everything yourself (Distills recommands using D3.js)
- Most of us are not used to Javascript, so making advanced interactive figures can be difficult at first
- Consequence: writing a Distill article takes a lot of time (In almost 5 years, fewer than 30 published papers)

Relations to IPOL 1/2

Distill and IPOL are not in direct competition:

- IPOL focuses on reproducibility (and the code) and complete explanation of what is done,
- Distill focuses on the communication, analysis of results.
- IPOL: article is usually not original research but rather linked to a main article.
- Distill paper must be either novel research, or analysis/review of existing research (ex: why some methods work/ why such artefacts exist)
- Distill: usually toy data in the demos, rather than real examples

Relations to IPOL 2/2

But we can learn from one another

- Concept: the demo is the article, not a separate thing
- Flexibility to change parametres and dissect both successes and failures

Can we take inspiration from them?

For who should IPOL articles and industrial applications be useful?

- The authors: To experiment on the method, find what works/doesn't work and why. They need **An infrastructure that makes their code more flexible and visualizable**.
- Other researchers: To reproduce the results, and to understand them. They need **Explanations of what is done, and clear visualizations of each step to understand it**.
- Lambda users: They only want to use the method and find its result. They usually don't care for the rest. They need **A simple interface to input data and see the result**.
- Industrial customers: They want a **showcase of the method** and a way to **use them in large quanitities**.

All the green needs can be made in one.

Suggestion for a new demo system

- The demo is the article, in other words explanations are within the demos, demos within the article (as figures of a pdf).
- One main end-to-end demo, then partial ones for each algorithm.
- Once you select the main input, all partial demos are synced to it, reversely changing parametres for one demo changes all those that follow

Proposed structure

- Main demo: Very similar to current demos, at the top of the article, but can be accessed at all times to globally change inputs (maybe minimized at the side/top of the page once you scroll down the article)
- Algorithmic part: usually, for each algorithm, one demo. All demos are synced.
- Experiments: Several images/sets of parametres are provided with comments and sample results, clicking on one sets the all demos on those inputs to see the whole workflow.

Algorithmic part

- Next to the demo is the pseudo-code and/or legible code in an easy-to-read language, that can easily be understood even for those not proficient with the language in question.
- When possible/relevant: the demo also shows the intermediate results of the algorithm, even if they are not shown in the main demo
- By default, input parametres come from the main demo, going in the natural flow of the method
- Parametres can be tuned in the demo, arbitrary inputs can also be injected instead of using inputs from previous algorithms
- When there is a change: update all the following algorithms, save to the archive
- Depending on the algorithm, demo could be a simple gallery (as current demos), but also a diagram or interactive figure showing the results

Algorithmic part

- Each demo/algorithm has a ”goto” label in the code, before the first call to this algorithm in the main method. Changing parametres restarts the full method from this label (no need to restart from the beginning to change something near the end of the method)
- Depending on resources and complexity of the algorithm: update real-time or on clicking a button? Update all subsequent algorithms right away, or only as the article is scrolled down?
- Save all values at each label for easier changes (for RAM-intensive variables, can be made common between several labels if they are not changed in-between)
- Needs some reflection for algorithms that are part of a loop
- Will yield many archive results for the same image with different parametres. Group them? Support for partial input?

Three modes

Choose from three modes:

- Default mode: You see the article, and all the demos within it. Useful to understand how the method works as well as do some simple experiments,
- Interface mode: The article is hidden, you see all the demos. Useful for more thorough experiments,
- Simple mode: Simplified demo, you provide the input and can possibly tune a few parametres, you receive the relevant output. For those who only wish to use the demo, not understand the article.

Writer: creates the demo and place them in the article for default mode. Interface and simplified mode can be made semi-automatically from it (you just have to select which parametres and outputs are relevant for the simple mode)

Issues

- Will probably take longer to create than current demos, but should also be more flexible and informative
- Probably requires a full, or almost full, restart on what we currently have
- Which framework?
  - Distill uses HTML+JS for its demos (they recommend D3.js but do not provide additionnal tools to use it)
    - Probably a lot of work to start from there for the IPOL team
    - Interaction with existing code?
    - I can't think of any advantages to use it, but mainly because I am not proficient in JS.
  - Jupyter notebook with Voilà should be able to do all that was discussed.
    - There should be fewer things to code from scratch.
    - Most people in data science/image processing are already familiar with Python (and notebooks can be used with other languages)
    - Easy integration with the code, but also with HTML.
    - Many existing widgets for easier demo creation, but it's also possible to create new ones with JS