CSCI 1377
Tools for Thought

Programming II
Notebooks

"It's been a source of ongoing bafflement and consternation for the past 29 years, that with the exception of a few people who get it, the community at large hasn't really adopted it, It's incalculable, literally … how much is lost, and how much time is wasted, and how many results are misinterpreted or are misrepresented."

— Theodore Gray, inventor of the Mathematica interface (2018)

Spreadsheets show the data and hide the code

D6 | =SUM(D2:D4)

	A	B	C	D
1	**Item**	**Unit Price**	**Quantity**	**Total Price**
2	Bacon	$11	2	$24
3	Eggs	$8	4	$35
4	Cheese	$5	10	$54
5				
6	**Sales tax**		*Subtotal*	$112
7	0.08			

IDEs show the code and hide the data

```
526  async fn create_reference_solution(
527      State(state: Arc<Mutex<AppState>>): State
528      Path(query: ReferenceSolutionQuery): Path
529  ) -> Result<Json<PullRequest>> {
530      let mut state: MutexGuard<'_, AppState> =
531      let ReferenceSolutionQuery {
532          quest_id: i64,
533          chapter_id: Option<usize>,
```

Spreadsheets show the data and hide the code

Notebooks show (some of) the data and (some of) the code

IDEs show the code and hide the data

D6 =SUM(D2:D4)

	A	B	C	D
1	**Item**	**Unit Price**	**Quantity**	**Total Price**
2	Bacon	$11	2	$24
3	Eggs	$8	4	$35
4	Cheese	$5	10	$54
5				
6	**Sales tax**		*Subtotal*	$112
7	0.08			

```
[17]: data = simulations.simulate_many(ExpandingSchedule, N=
      ax = sns.lineplot(data, x="time", y="recall_prob", est
      _ = ax.set_ylim([-0.05, 1.05])
```


```
526  async fn create_reference_solution(
527      State(state: Arc<Mutex<AppState>>): State
528      Path(query: ReferenceSolutionQuery): Path
529  ) -> Result<Json<PullRequest>> {
530      let mut state: MutexGuard<'_, AppState> =
531      let ReferenceSolutionQuery {
532          quest_id: i64,
533          chapter_id: Option<usize>,
```

Notebooks are about...

Computational exploration

REPLs, shells, scratchpads

Scientific process

Computational narrative

Literate programming

Scientific communication

Computational exploration

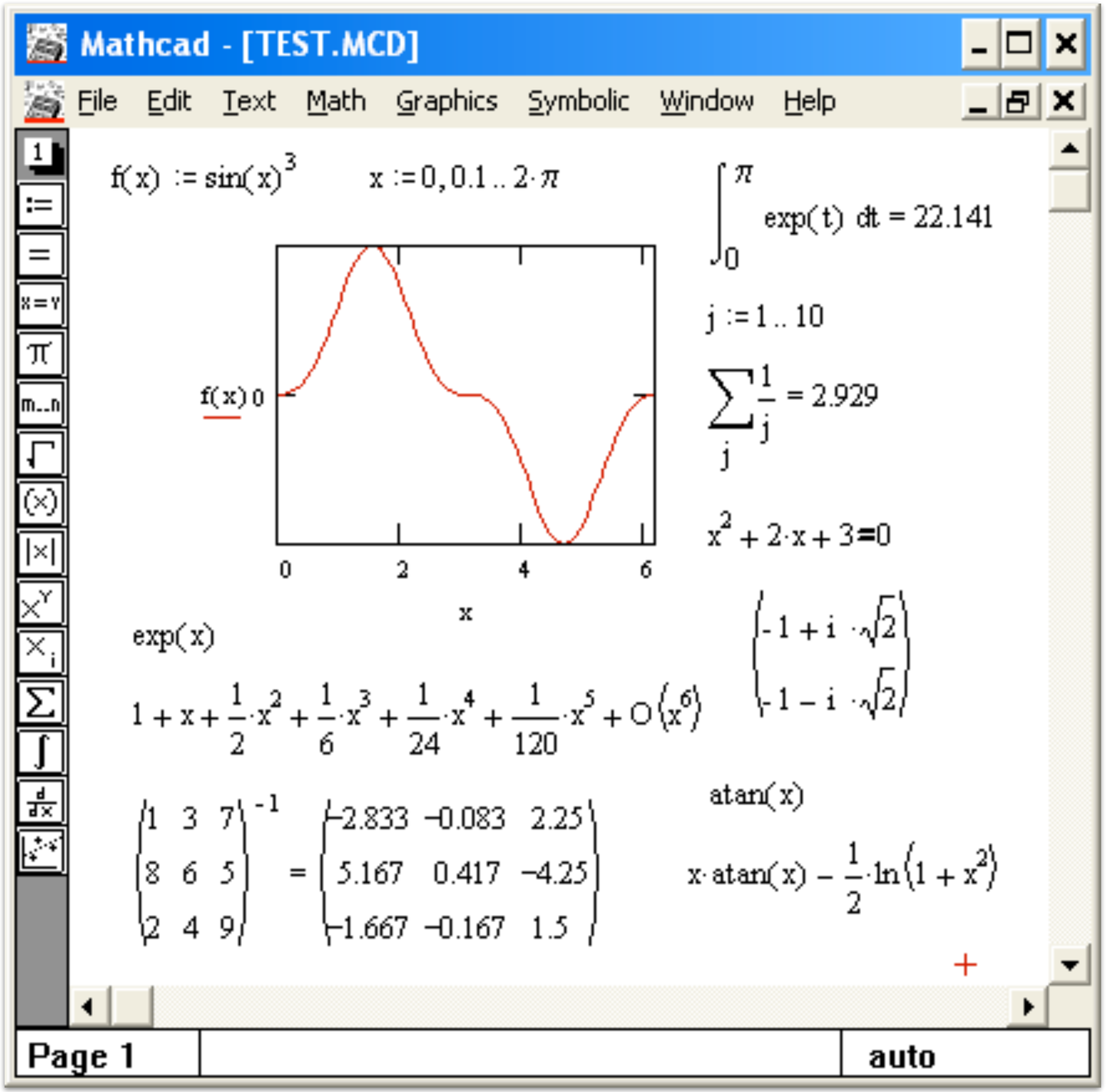
Read
Eval
Print
Loop

```
(loop
  (print
    (eval
      (read))))
```

```
~ ››› python3
Python 3.14.2 (main, Dec  5 2025, 16:49:16)
(clang-1700.4.4.1)] on darwin
Type "help", "copyright", "credits" or "lic
information.
>>> import json
>>> data = [{"foo": "bar"}]
>>> print(json.dumps(data))
[{"foo": "bar"}]
```

Mathematical notebooks caught fire in the late 80s

MathCAD.™ The first software that lets you do calculations on your PC as simply as on a scratchpad. Just define your variables and enter your formulas anywhere on the screen. MathCAD not only formats your equations as they're typed, it instantly calculates the results, and displays your work in real math notation.

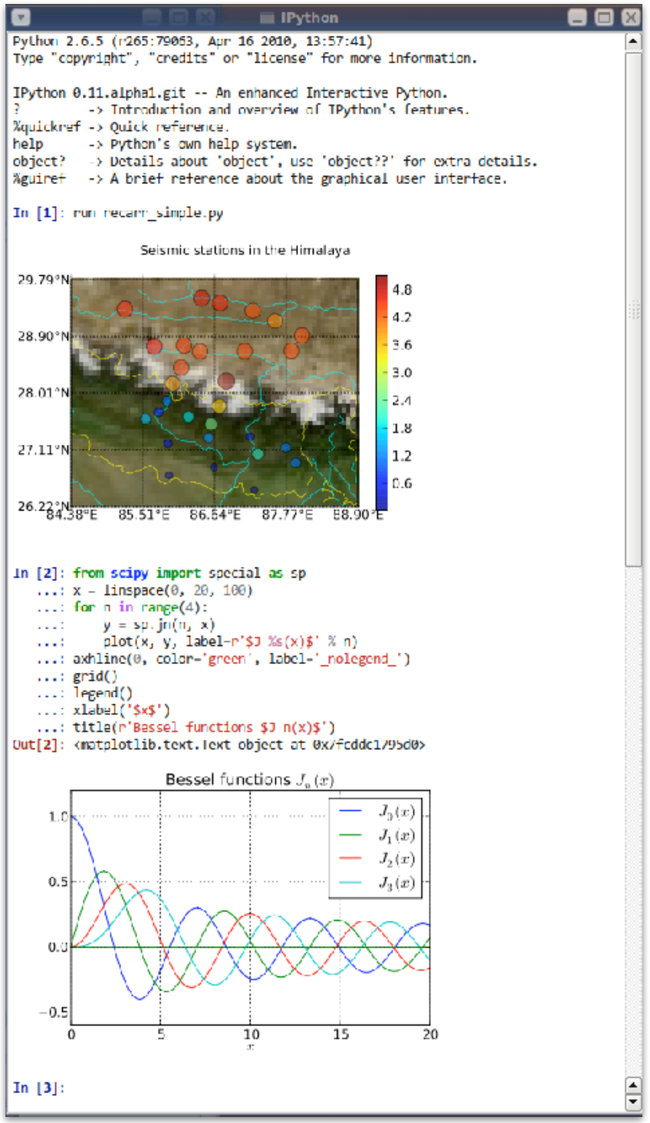


[Mathematica demo]

IPython was born as Mathematica for Python

"I was a graduate student in physics at CU Boulder [...] I was a heavy user of the Mathematica notebooks and liked them a lot. I started using Python in 2001 and liked the language, but its interactive prompt felt like a crippled toy compared to the systems mentioned above or to a Unix shell. [...] I then wrote a python startup file to provide [capturing results] and some other niceties such as loading Numeric and Gnuplot, giving me a 'mini-mathematica' in Python."

Fernando Perez



Fernando Perez. "The IPython notebook: a historical retrospective" 2012

[IPython demo]

IPython got a web-based notebook in 2011

IPython Notebook

Spectrogram Save Idle

Notebook

Actions New Open

Download ipynb Print

Cell

Actions Delete

Format Code Markdown

Output Toggle ClearAll

Insert Above Below

Move Up Down

Run Selected All

Autoindent:

Kernel

Actions Interrupt Restart

Kill kernel upon exit:

Help

Links Python IPython

NumPy SciPy

MPL SymPy

Shift-Enter : run selected cell

Ctrl-Enter : run in terminal mode

Ctrl-m h : show keyboard shortcuts

Simple spectral analysis

An illustration of the Discrete Fourier Transform

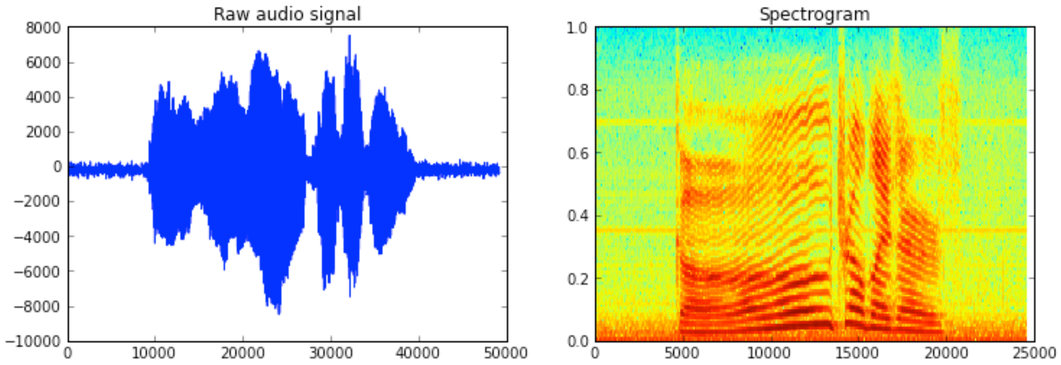
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

using windowing, to reveal the frequency content of a sound signal.
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('/home/fperez/teach/py4science/book/examples/test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```



Fernando Perez. “The IPython notebook: a historical retrospective” 2012

IPython became Jupyter in 2014

What's in a name?

- *Inspired* by the open languages of science:
  - Julia, Python & R
  - *not* an acronym: *all languages* equal class citizens.
- Astronomy and Scientific Python:
  - A long and fruitful collaboration
- Galileo's notebooks:
  - the original, open science, data-and-narrative papers
  - Authorea: "Science was Always meant to be Open"

Notebooks as tools for computational exploration

- Interactive programming allows for incremental construction and execution of programs
  - Originally with REPL, then later enhanced REPLs (e.g. IPython), finally notebooks
  - Keep values in memory while programming!!
  - Close connection to dynamically typed languages

- Code cells enable co-location of code and rich outputs
  - Text, plots, images, interactive widgets

- Auxiliary tools for managing notebook structure
  - Collapsing code, entire sections

Computational narrative

Literate programming centers text over code

```
@ This program has no input, because
we want to keep it rather simple.
[...]

\[The program text below specifies the
''expanded meaning'' of [...]\]

@<Program to print...@>=
program print_primes(output);
const @!m=1000;
@<Other constants of the program@>@;
var @<Variables of the program@>@;
begin @<Print the first |m| prime
numbers@>;
end.
```

Weave →

2. This program has no input, because we want to keep it rather simple. The result of the program will be to produce a list of the first thousand prime numbers, and this list will appear on the *output* file.

Since there is no input, we declare the value $m = 1000$ as a compile-time constant. The program itself is capable of generating the first m prime numbers for any positive m, as long as the computer's finite limitations are not exceeded.

⟦The program text below specifies the "expanded meaning" of ⟨ Program to print ... numbers 2 ⟩; notice that it involves the top-level descriptions of three other sections. When those top-level descriptions are replaced by their expanded meanings, a syntactically correct PASCAL program will be obtained.⟧

⟨ Program to print the first thousand prime numbers 2 ⟩ ≡

```
program print_primes(output);
  const m = 1000;
    ⟨ Other constants of the program 5 ⟩
  var ⟨ Variables of the program 4 ⟩
    begin ⟨ Print the first m prime numbers 3 ⟩;
    end.
```

↓ Tangle

```
program print_primes(output);
const m=1000; rr=50; cc=4; ww=10;
var p:array[1..m] of integer;
```

Donald Knuth. "Literate Programming" 1984

Physically Based Rendering

§1. Verbs. "Booting" is the traditional computing term
switches on it has no program to run, but to load in a
having a minimal "boot" program wired into the hardw

So too with Inform. The opening sentence of the Basic

The verb to mean means the meaning relation.

(See Preamble (in basic_inform).) But this is circular: if
the verb, or know what it means? We break this circula

```
void BootVerbs::make_built_in(void) {
    verb *to_mean;
    special_meaning_holder *meaning_of_m
    Create the special meanings 1.3;
    Create the verbs to be and to mean 1.5
    Give meaning to mean 1.6;
}
```

Inform7 compiler

It is also useful to be able to compute XYZ coefficients for a SampledSpectrum. Because SampledSpectrum only has point samples of the spectral distribution at predetermined wavelengths, they are found via a Monte Carlo estimate of Equation (4.22) using the sampled spectral values s_i at wavelengths λ_i and their associated PDFs:

$$x_\lambda \approx \frac{1}{\int_\lambda Y(\lambda)\,\mathrm{d}\lambda}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{s_i\,X(\lambda_i)}{p(\lambda_i)}\right),$$

(4.23)

and so forth, where n is the number of wavelength samples.

SampledSpectrum::ToXYZ() computes the value of this estimator.

<<Spectrum Method Definitions>>+=

```
XYZ SampledSpectrum::ToXYZ(const SampledWavelengths &lambda) const {
    <<Sample the X, Y, and Z matching curves at lambda>> ⊞
    <<Evaluate estimator to compute (x, y, z) coefficients>> ⊞
}
```

The first step is to sample the matching curves at the specified wavelengths.

<<Sample the X, Y, and Z matching curves at lambda>>=

```
SampledSpectrum X = Spectra::X().Sample(lambda);
SampledSpectrum Y = Spectra::Y().Sample(lambda);
SampledSpectrum Z = Spectra::Z().Sample(lambda);
```

Jupyter was seen as a new kind of literate programming

> “We therefore refer to the workflow exposed by computational notebooks as "literate computing": it is the weaving of a narrative directly into a live computation, interleaving text with code and results to construct a complete piece that relies equally on the textual explanations and the computational components. For the goals of communicating results in scientific computing and data analysis, I think this model is a better fit than the literate programming one, which is rather aimed at developing software in tight concert with its design and explanatory documentation.”

Fernando Perez. “"Literate computing" and computational reproducibility: IPython in the age of data-driven journalism” 2013

LIGO gravitational waves notebook

Plot the Amplitude Spectral Density (ASD)

Plotting these data in the Fourier domain gives us an idea of the frequency content of the data. A way to visualize the frequency content of the data is to plot the amplitude spectral density, ASD.

The ASDs are the square root of the power spectral densities (PSDs), which are averages of the square of the fast fourier transforms (FFTs) of the data.

They are an estimate of the "strain-equivalent noise" of the detectors versus frequency, which limit the ability of the detectors to identify GW signals.

They are in units of strain/rt(Hz). So, if you want to know the root-mean-square (rms) strain noise in a frequency band, integrate (sum) the squares of the ASD over that band, then take the square-root.

There's a signal in these data! For the moment, let's ignore that, and assume it's all noise.

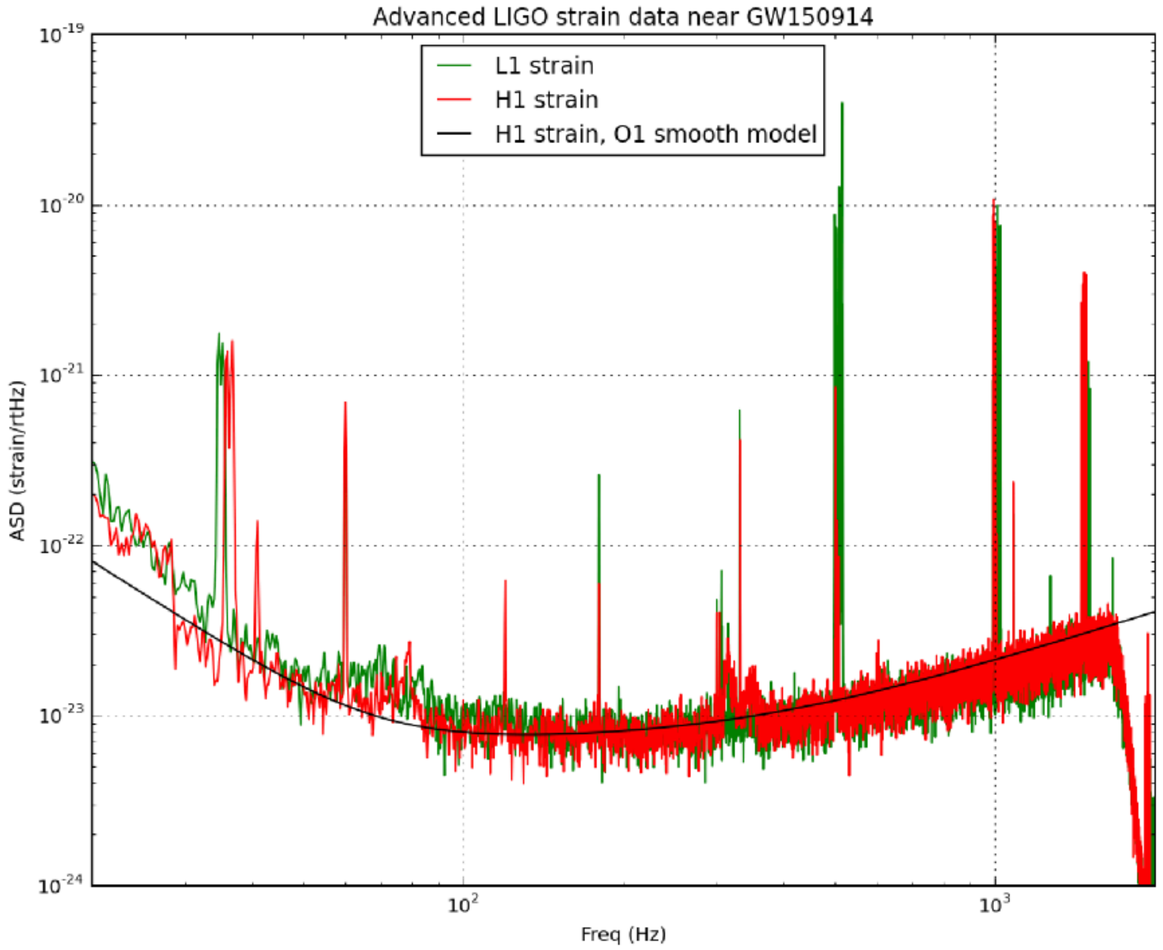
In [8]:

```
make_psds = 1
if make_psds:
    # number of sample for the fast fourier transform:
    NFFT = 4*fs
    Pxx_H1, freqs = mlab.psd(strain_H1, Fs = fs, NFFT = NFFT
    Pxx_L1, freqs = mlab.psd(strain_L1, Fs = fs, NFFT = NFFT

    # We will use interpolations of the ASDs computed above
    psd_H1 = interp1d(freqs, Pxx_H1)
    psd_L1 = interp1d(freqs, Pxx_L1)

    # Here is an approximate, smoothed PSD for H1 during O1,
    Pxx = (1.e-22*(18./(0.1+freqs))**2)**2+0.7e-23**2+((freq
    psd_smooth = interp1d(freqs, Pxx)

if make_plots:
    # plot the ASDs, with the template overlaid:
    f_min = 20.
```



Notebooks as tools for computational narrative

- Literate programming laid the foundation for interleaving text and code
  - Focus on (a) nested, named code holes with (b) separate code / doc outputs
- Notebooks provide text cells for rich text commentary
  - Mathematica provides nested structure, Jupyter is linear
- In theory, notebooks are a better rhetorical technology for technical communication

Notebooks in the modern day

SCIENCE

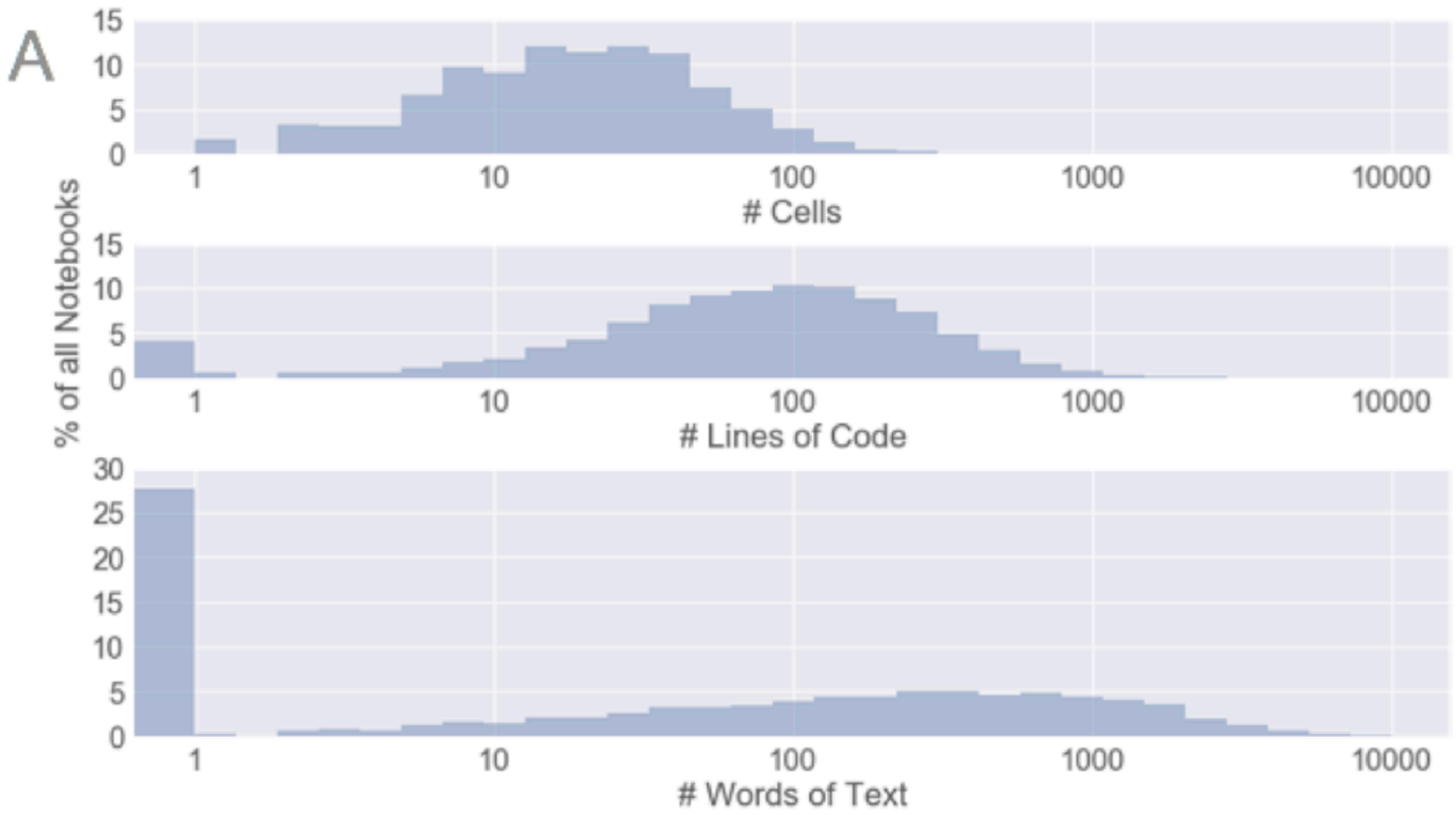
THE SCIENTIFIC PAPER IS OBSOLETE

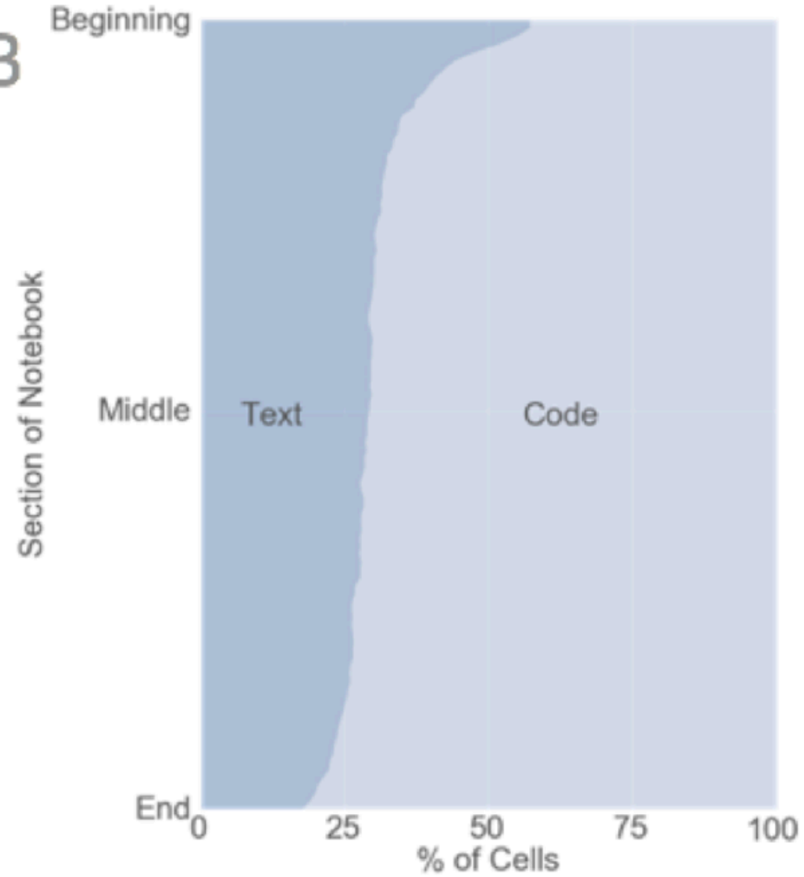
Here's what's next.

By James Somers

“Anyone could inspect [IPython] code and modify it, contributing their changes back to the common cause. It was a deliberate decision. “I was interested in both the ethical aspect of being able to share my work with others,” Pérez told me—he came from Colombia, where access to proprietary software was harder to come by—“and there was also a more epistemological motivation.” He thought that if science was to be an open enterprise, the tools that are used to do it should themselves be open. Commercial software whose source code you were legally prohibited from reading was “antithetical to the idea of science,” where the very purpose is to open the black box of nature.”

Notebooks are usually more code than text





I could find zero documented cases of a published paper generated directly from a Jupyter notebook!

Rule, Tabard, and Hollan. “Exploration and Explanation in Computational Notebooks” 2018

Notebooks are dubiously reproducible

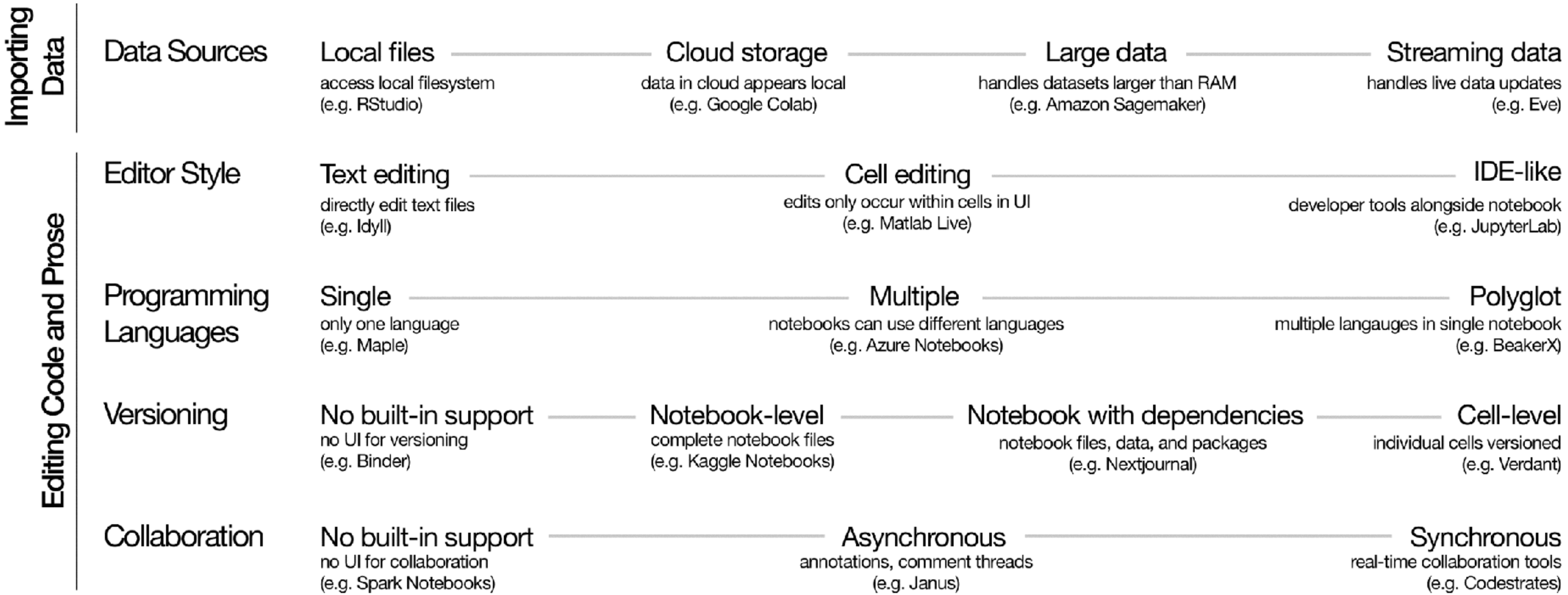
"Out of 27,271 Jupyter notebooks from 2,660 GitHub repositories associated with 3,467 publications, 22,578 notebooks were written in Python, including 15,817 that had their dependencies declared in standard requirement files and that we attempted to re-run automatically. For 10,388 of these, all declared dependencies could be installed successfully, and we re-ran them to assess reproducibility. Of these, 1,203 notebooks ran through without any errors, including 879 that produced results identical to those reported in the original notebook, and 324 for which our results differed from the originally reported ones. Running the other notebooks resulted in exceptions."

879 / 27,271 = 3%!!

Samuel and Mietchen. "Computational reproducibility of Jupyter notebooks from biomedical publications" 2023

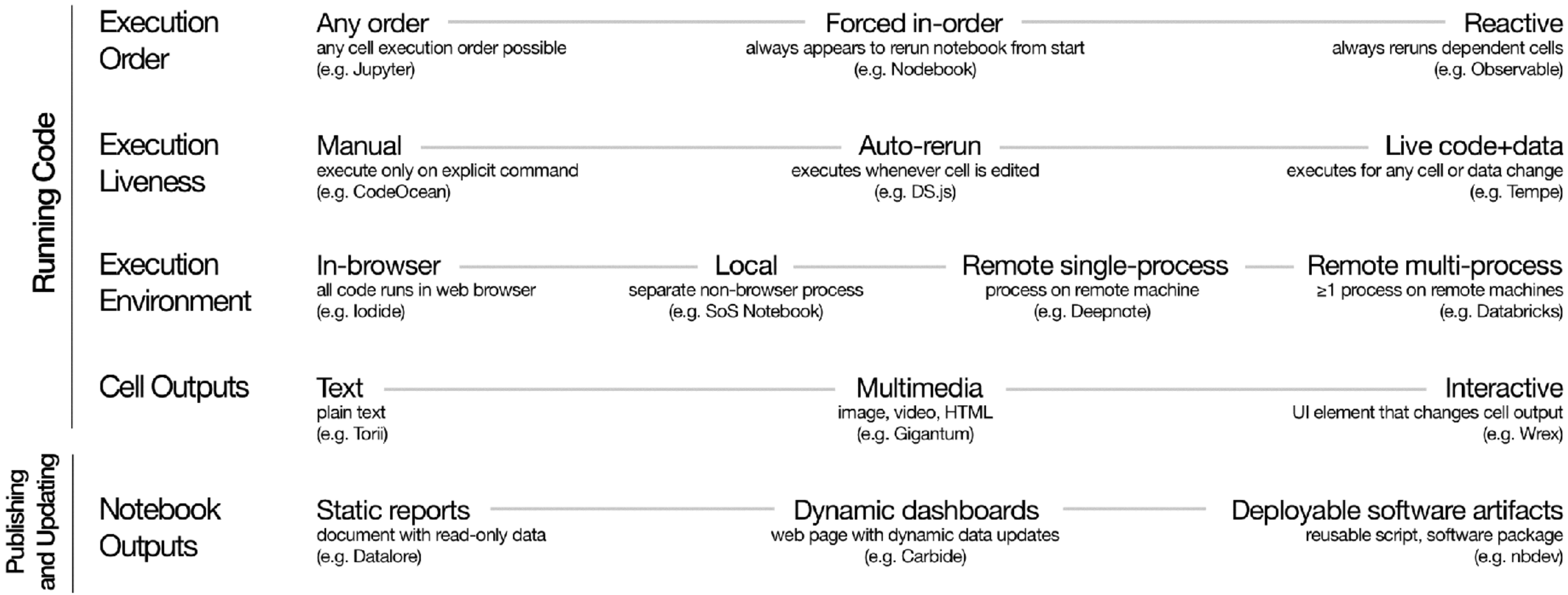
The future of notebooks

Design dimensions of computational notebooks



Lau et al. “The Design Space of Computational Notebooks: An Analysis of 60 Systems in Academia and Industry” 2020

Design dimensions of computational notebooks



Lau et al. "The Design Space of Computational Notebooks: An Analysis of 60 Systems in Academia and Industry" 2020

Reactive notebooks auto-execute cells (sometimes)

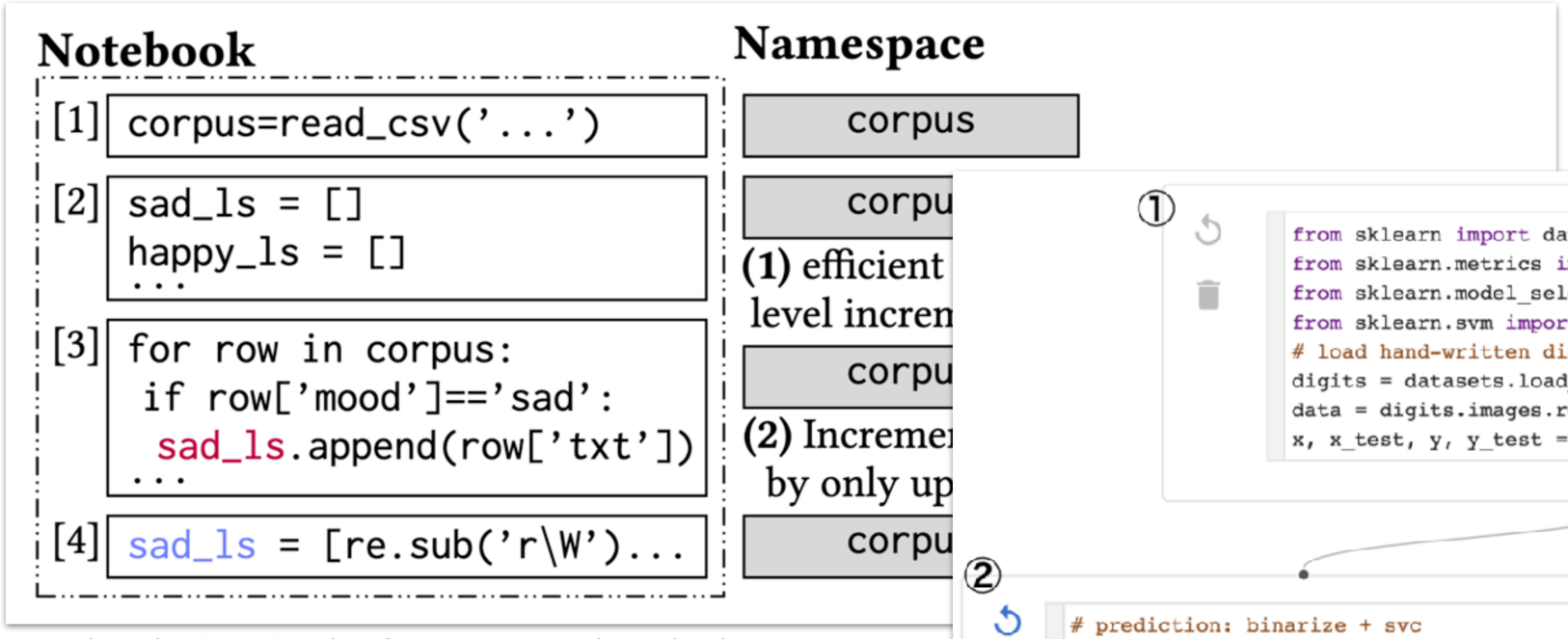
Front page:

"marimo guarantees your notebook code, outputs, and program state are consistent. This solves many problems associated with traditional notebooks like Jupyter."

Deep in the docs:

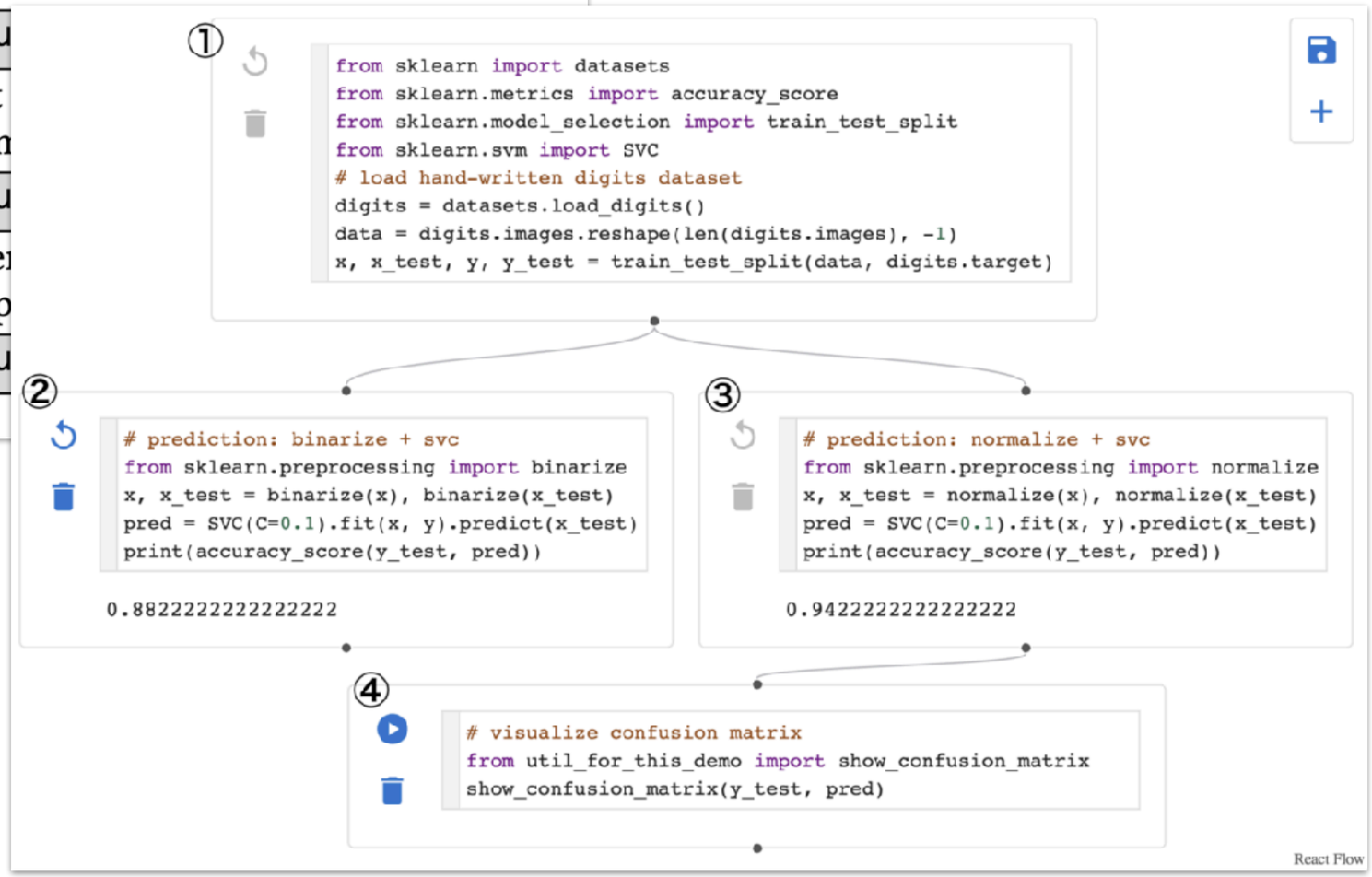
"marimo does not track mutations to objects, *e.g.*, mutations like `my_list.append(42)` or `my_object.value = 42` don't trigger reactive re-runs of other cells. Tracking mutations reliably is impossible in Python."

Time-traveling notebooks improve revision



Li et al. "Kishu: Time-Traveling for Computational Notebooks" 2024

Sato and Nakamaru. "Multiverse Notebook: Shifting Data Scientists to Time Travelers" 2024



Narrative is moving towards Markdown++

Quarto

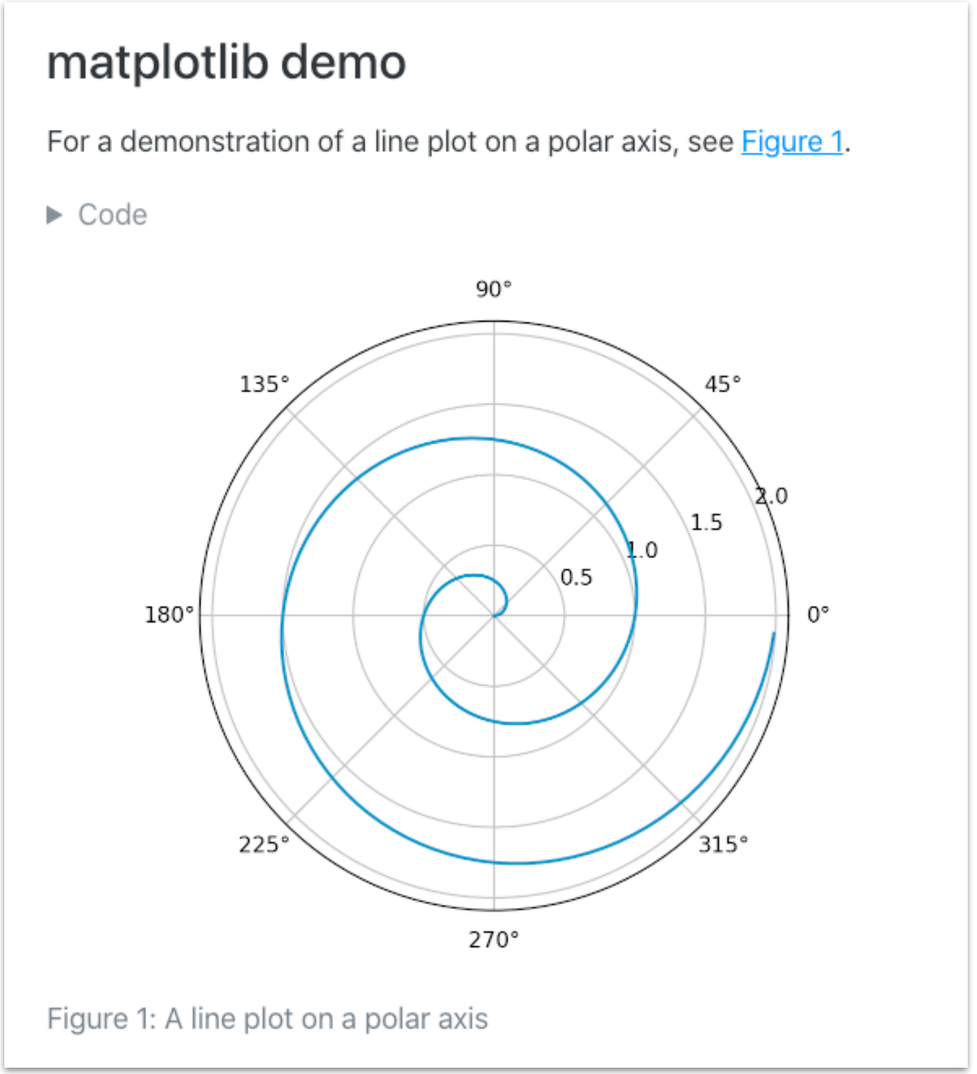
````
---
title: "matplotlib demo"
format:
  html:
    code-fold: true
jupyter: python3
---

For a demonstration of a line plot on a polar
axis, see @fig-polar.

```{python}
#| label: fig-polar
#| fig-cap: "A line plot on a polar axis"

import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
# ...
plt.show()
```
````



Narrative is moving towards Markdown++

Living Papers

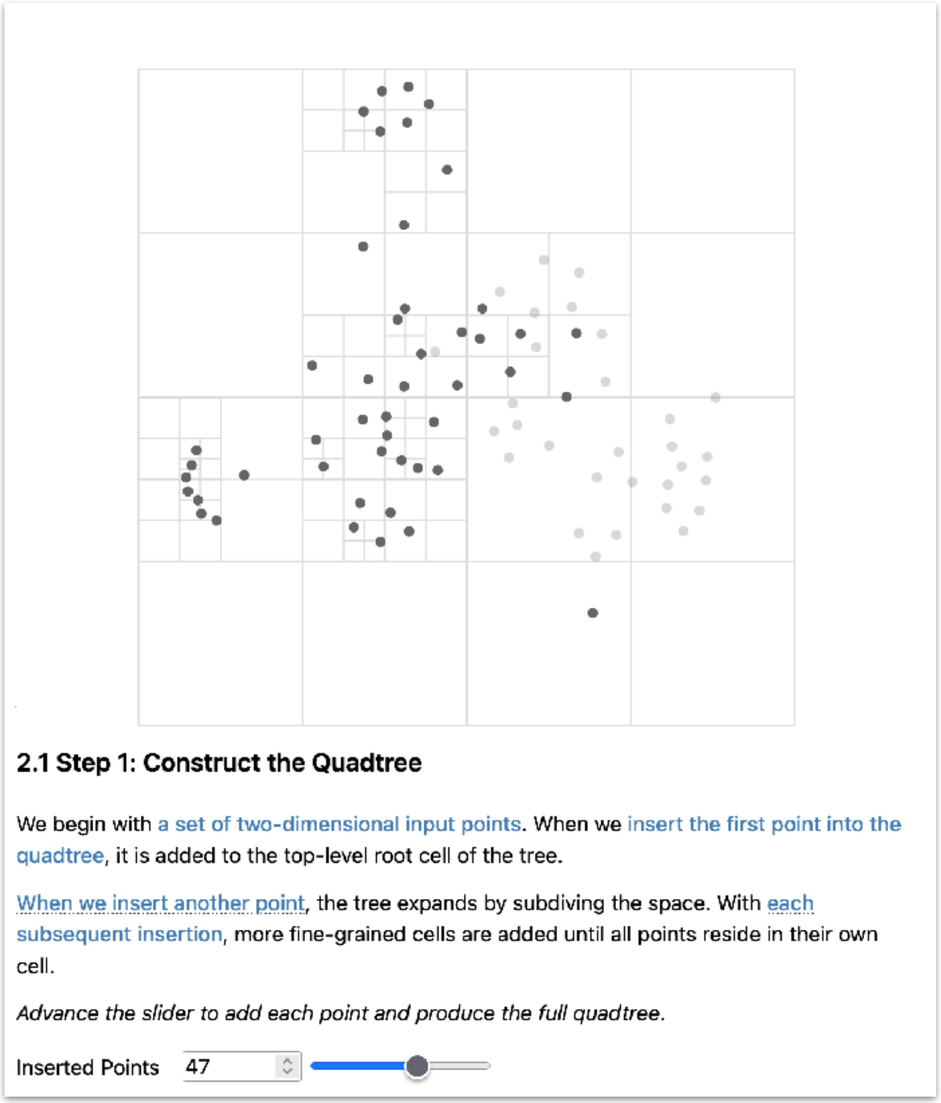
```
## Step 1: Construct the Quadtree

We begin with [a set of two-dimensional input points]
(`layout=false, step=0`). [...]

With
[each](`layout=false, step=Math.min(step + 1, 77)`)
[subsequent](`layout=false, step=Math.min(step + 1, 77)`)
[insertion](`layout=false, step=Math.min(step + 1, 77)`),
more fine-grained cells are added until all points reside in
their own cell.

_Advance the slider to add each point and produce the full
quadtree_.

~~~ js {bind=step}
Inputs.range([0, 77], { step: 1, label: 'Inserted Points' })
~~~
```



Narrative is moving towards Markdown++

NOTA is a language for writing documents, like academic papers and blog posts. The goal of **NOTA** is to **bring documents into the 21st century.**

Documents contain a lot of structure — for example, "**NOTA**" is a reference to a term defined in the preceding paragraph. **NOTA** enables authors to *represent* that structure, which allows the reading medium (the browser) to *understand* that structure, which in turn empowers readers to *use* that structure. Try clicking on any "**NOTA**" reference to see this idea in action. (Then try double-clicking!)

A **NOTA** document compiles to a JavaScript program, meaning it's easy to:

- View documents on any device that has a web browser.
- Use variables, functions, and data structures to simplify document writing.
- Integrate with JavaScript libraries like KaTeX, Vega-Lite, and Penrose.
- Support accessibility needs like screen readers.

Show JS

B I U @

```
// This code is editable!
% let nota = @Smallcaps{**Nota**}
.@Definition[name: "nota", label: nota]{
  #nota is a language for writing documents, like academic papers and blog
  posts.
}
The goal of &nota is to **bring documents into the #(20 + 1)st century.**

Documents contain a lot of structure --- for example, "&nota" is a reference to
a term defined in the preceding paragraph. &nota enables authors to *represent*
that structure, which allows the reading medium (the browser) to *understand*
that structure, which in turn empowers readers to *use* that structure. Try
clicking on any "&nota" reference to see this idea in action. (Then try double-
clicking!)

A &nota document compiles to a JavaScript program, meaning it's easy to:

* View documents on any device that has a web browser.
* Use variables, functions, and data structures to simplify document writing.
* Integrate with JavaScript libraries like [KaTeX](https://katex.org/), [Vega-
Lite](https://vega.github.io/vega-lite/), and [Penrose](https://github.com/pen-
rose/penrose).
* Support accessibility needs like screen readers.
```