FLUID

Data visualisation for the 21st century

or

“Choose your own data-viz adventure”

Roly Perera[1, 2] & Achintya Rao[1, 3]

[1] Institute of Computing for Climate Science, University of Cambridge
[2] School of Computer Science, University of Bristol
[3] Science Communication Unit, UWE Bristol

Plan for rest of session

Fluid is a **functional programming language** for building **transparent, self-explanatory** research outputs

- Project overview
- Demos & UX discussion
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- Call for Collaboration!

Plan for rest of session

Fluid is a **functional programming language** for building **transparent, self-explanatory** research outputs

- Project overview
- Demos & UX discussion
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- Call for Collaboration!

- **Project overview**
- Demos & UX discussion
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- Call for Collaboration!

Project overview

Initial funding from The Alan Turing Institute
Now collaboration between UoB and ICCS

Contributors (past & present)

Joe Bond [1]
Colin Crawford [4]
Cristina David [1]
Harleen Gulati [1]
Hana Iza Kim [2]
Minh Nguyen [1]
Dominic Orchard [5, 2]
Roly Perera [2, 1]
Tomas Petricek [3]
Achintya Rao [6, 2]
Meng Wang [1]

[1]University of Bristol
[2]University of Cambridge
[3]Charles University
[4]University of Edinburgh
[5]University of Kent
[6]University of West of England

https://f.luid.org
https://github.com/explorable-viz/fluid

- Project overview
- **Demos & UX discussion**
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- Call for Collaboration!

Demo: renewables

Demo: renewables

Ideas introduced

- Surfacing fine-grained I/O relationships
  — different parts of the output use different parts of the input!
- The more output we express interest in, the more data becomes relevant
- Transient/persistent selections allow **relative** queries

Discussion points

- You could program these “transparency queries” yourself — but not scalable or robust to change
  — to be **ubiquitous** needs to be **automatic** (more on this later)

Demo: renewables

```
let series type country = [
   { x: row.year, y: row.output }
   | year ← [2013..2018], row ← renewables,
   row.year == year, row.energyType == type, row.country == country
] in LineChart {
   caption: "Output of USA relative to China",
   plots: [
      LinePlot { name: type, data: plot }
      | type ← ["Bio", "Hydro", "Solar", "Wind"],
      let plot = zipWith (fun p1 p2 → { x: p1.x, y: p1.y / p2.y })
                         (series type "USA") (series type "China")
   ]
}
```

Demo: renewables-linked

Demo: renewables-linked

Ideas introduced

- Surfacing relationships between outputs that arise from common data dependencies
  — comprehension aid for understanding how different views are related
- “Brushing and linking”, but **transparent**: shared data explains why two selections are related
  — relative queries can reveal how data usage “overlaps”
- Additional dimension of selection: **primary** vs. **secondary**

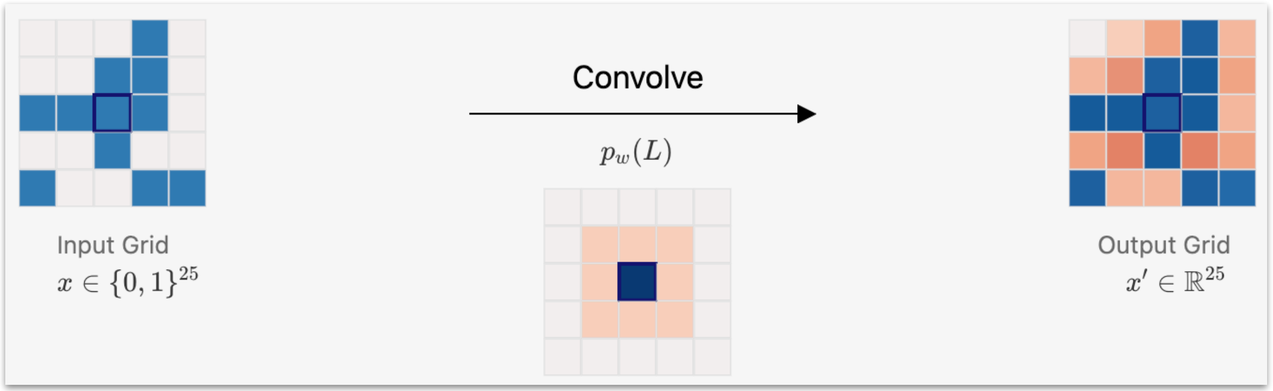
Discussion points

- Visualising selections: non-trivial and visualisation-dependent
  — cognitive considerations like change blindness/inattentional blindness
- Just knowing **what** is related isn’t enough — need to know **how**

Future directions

- “Intensional” queries — how was this calculated?

Demo: convolution



Demo: convolution

Ideas introduced

- Views of computations can be used to explain and illustrate
  — infrastructure for “explorable explanations” (Bret Victor)
- Interacting with **inputs** to see what **outputs** they contribute to
- Inputs can be related in virtue of contributing to a common output

Discussion points

- distill.pub example was a **hand-crafted animation**
- Fluid demo is a **transparent algorithm**
  — generic infrastructure that can be reused for other algorithms
  — infrastructural improvements shared by everything downstream

Demo: convolution

```
let zero n = const n;
    wrap n n_max = ((n - 1) `mod` n_max) + 1;
    extend n = min (max n 1);

let convolve image kernel method =
    let ((m, n), (i, j)) = (dims image, dims kernel);
        (half_i, half_j) = (i `quot` 2, j `quot` 2);
        area = i * j
    in  [| let weightedSum = sum [
              image!(x, y) * kernel!(i' + 1, j' + 1)
              | (i', j') ← range (0, 0) (i - 1, j - 1),
                let x = method (m' + i' - half_i) m,
                let y = method (n' + j' - half_j) n,
                x ⩾ 1, x ⩽ m, y ⩾ 1, y ⩽ n
           ] in weightedSum `quot` area
           | (m', n') in (m, n) |];
```

Demo: convolution-wrapped

Ideas introduced

- Comprehension as active process of hypothesis formulation and testing
- Interact with different algorithms to understand how they behave

Discussion points

- Implementation details start to matter in new ways!

Future directions

- “Unpack” computation to examine intermediate results (e.g. interim 3x3 arrays)
- Converge the medium in which we **do** science with medium in which we **communicate** science

- Project overview
- Demos & UX discussion
- **Creating content with Fluid**
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- Call for Collaboration!

Creating a Fluid visualisation

So far project has focused mainly on **end-user experience**
Developer tools are sorely lacking..

Current workflow (😬)

- Create `index.html` with `div` to contain Fluid visualisation
- Write Fluid source code (`.fld`) for visualisation
- Write small PureScript program to load visualisation

What's missing

- Command-line publishing tool (Node.js)
- Fluid kernel for Jupyter — will enable authoring via **Markdown** and **Quarto**

Want to leverage as much existing open source infrastructure as possible
Three interns will be helping with some of this over the summer!

Language overview

Current design

- Purely functional (no side-effects)
- Untyped
- Records, lists, dictionaries, 2D arrays
- Graphics library based on **d3.js**

Implemented in **PureScript**
(Haskell clone for the web)

What's missing

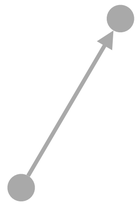
- Modules and imports
- I/O — load from file, db or URL
- User-defined datatypes
- Type system (with units of measure?)

- Project overview
- Demos & UX discussion
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- **Behind the scenes: dynamic dependence graphs**
- Future directions
- Call for Collaboration!

How does it work?

Key infrastructure

- Fluid interpreter builds a **directed acyclic graph** of data dependencies as program runs
- Treat relationship between inputs and outputs as “metadata” that can be queried
- Related inputs and related outputs are relations of **cognacy** (common ancestry) in the graph



Further reading

- *Conjugate Operators for Transparent, Explorable Research Outputs.* Bond, David, Nguyen, Orchard & Perera, 2024
- *Linked Visualisations via Galois Dependencies.* Perera, Nguyen, Petricek & Wang, 2022
- *Provenance for Interactive Visualizations.* Psallidas & Wu, 2018

- Project overview
- Demos & UX discussion
- Creating content with Fluid
  - authoring experience today
  - where we want to take it
  - language overview
- Behind the scenes: dynamic dependence graphs
- **Future directions**
- Call for Collaboration!

Future directions

How to enable a smooth transition from content consumption to content creation?

readers  authors

Readers:

- Don’t care how it works
- Want responsive, self-explanatory, intuitive UI
- Should be able to transition from passive reading to active engagement
- UI affordances (opportunities for interaction) should present themselves

We wear different hats at different times..

Authors:

- Proficient in technology
- Invested in specific workflows and skills
- Benefits of new technology need to be obvious
- Barriers to entry need to be low

What are the prospects of doing something like Fluid for R or Python?

Weaving new stories from existing ones

Example of previous continuum that we would like to enable:

- Reviewer explores claims, data sources and computational methods **in situ**
- Frames queries by interacting with outputs and perhaps making other choices
- Queries/views are **persistent** and versioned and can be shared with authors or other readers
- Interactions are reproducible and can be replayed (cf. Histogram)
- Interesting observations are new knowledge and contribute to the overall science
- Authors' original narrative was just one of many possible narratives

Not a new idea, but definitely an idea whose time has come

- Project overview
- Demos & UX discussion
- Creating content with Fluid
  - — authoring experience today
  - — where we want to take it
  - — language overview
- Behind the scenes: dynamic dependence graphs
- Future directions
- **Call for Collaboration!**

How can we help you tell a story with your data?

We are looking for a VESRI project that we can use to showcase **our infrastructure** and **your research** in the form of an online article with figures transparently linked to data

Timeframe: Aug-Dec 2024

Call for Case Studies

Do you have:

- an existing open access publication with figures we could reimplement in Fluid to add value?
- a question about prior work that you didn't get a chance to ask and would like to revisit?
- a new idea or question we could help you present in an interesting way?

Your project may be a good fit if:

- not too data-intensive or algorithmically complex!
- has obvious visual elements
- will benefit from being presented in a transparent, explorable way (potentially to a lay audience)