# HoloViz: Streams Deep Dive*

***Concepts and history***

** but not streamz!*

ANACONDA.

# Some HoloViews History

# How it Started I

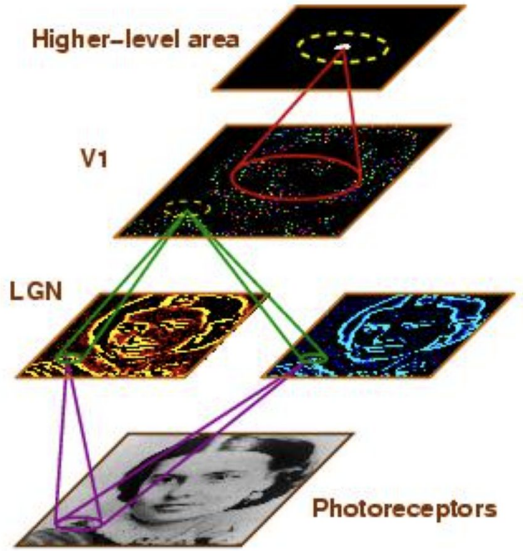
- Started my PhD in computational neuroscience at the University of Edinburgh in 2010
- Met Jim and Philipp during the Master's year of the program (Doctoral Training Center)
- Decided to investigate temporal processing in primary visual cortex (V1) for my PhD
- Jim agreed we needed to extend the visualization tools of our simulator Topographica.
- Started 'DataViews' as an extension of 'SheetViews' used for visualizing 2D numpy arrays (i.e. images)

# How it Started II

- IPython Notebook (later Jupyter) had just gone public (version 0.11 IIRC) and it looked like an interesting computation environment for research.
- Found the name “DataViews” to be horrible and unsearchable.
- While at an experimental lab in Marseille, a collaborator helped suggest the ‘holo’ prefix (see *holo*gram, *holi*stic, *whole*)
- Invented the `HoloMap` to allow browsing `Image`s over one or more dimensions.
- Philipp joined after about a year after I started wanting to avoid looking at giant PDF files of simulation results.
- Won an open source award in 2013 and released HoloViews 1.0 (Philipp still has the trophy)
- Finished my PhD and never thought about it again ;-p

# TOPOGRAPHICA

- Topographica is simulator of rate-based neural sheets, designed to build developmental simulations of the mammalian visual system.
- In many ways, the network is like a convolutional neural network for image processing (with biological constraints)
- Over development, the patterns of neural responses change due to changing synaptic (i.e. network) weights.
- Xarray and pandas didn't exist and a 'SheetView' class was used to visualize 2D numpy arrays.



# SheetViews

- Topographica had a GUI written in Tk with tkinter.
- It was already an old toolkit at the time (~2010) but it worked!
- Visualization was handled by a mixture of PIL and Matplotlib
- Matplotlib's API was even worse back then and even more like Matlab...
- The GridSpec API didn't exist, so laying out plots was very painful...
- Overlaying data was also painful, with multiple coordinate systems to think about.
- I just wanted to use `+` to put things next to each other and `*` to put things over each other.
- So HoloViews (DataViews) started with Images and the `+` and `*` operators.
- It was only possible to use these operators to build up compositional plots due to the new IPython Notebook environment.
- With display hooks, Image elements could visualize themselves in notebooks.

# HoloMaps

- Visualizing the activity and responses (e.g OR, DR) of neural sheets over time was a core part of my PhD.
- The idea of `HoloMap` was that it was a multi-dimensional dictionary of visualizable elements. One of those dimensions could be time but other dimensions also had to be supported.
- Initially time varying output was in the form of MPEGs and GIFs.
- A HoloMap was a much more declarative way to create animated output with matplotlib than the callback approach it supported.

# Python Evolves…

- Python became popular for some reason…
- Pandas is invented. What does Python need tabular data for anyway?
- Xarray is invented to support labelled arrays: while HoloViews was focused on visualization, adding dimensioned metadata was one of our core goals.
- Sadly still no standard, universal unit support for scientific Python :-(
- Matplotlib is already old and clunky but a library called ‘Bokeh’ appears…
- Bokeh works in notebooks and is interactive: maybe we can do better than GIFs?
- Philipp adds the first set of JavaScript widgets to HoloMaps
- Now we can use sliders to scrub through Images over one or more dimensions!
- Philipp adds the Bokeh backend

# The problem with Eddie...

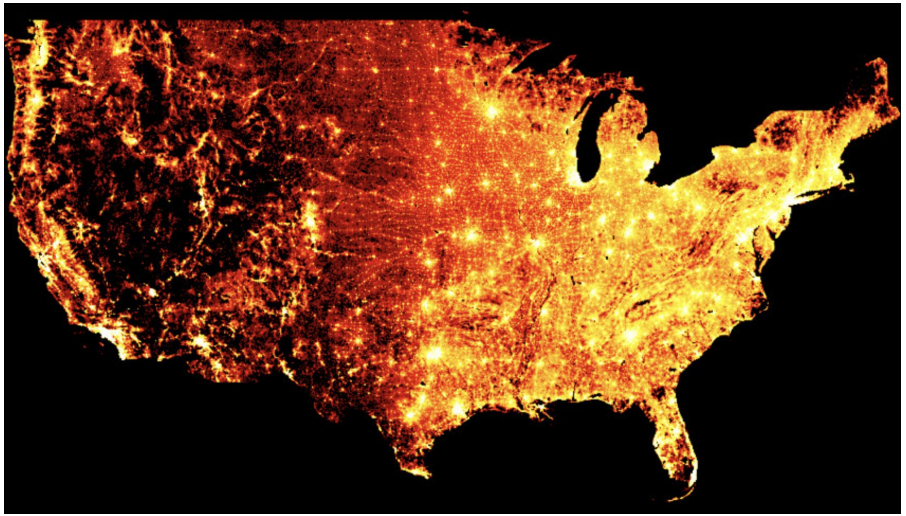
- Eddie is one of the HPC systems at the University of Edinburgh
- Our simulations would often take *days* to run...
- ...and we would often need to do large parameter searches
- Over each simulation run, topographica would dump out pickled Image objects.
- Building a `HoloMap` in the middle of a run involved collecting and loading all these `Image` objects into memory.
- What if HoloMap were less like a static collection but more like a dictionary where a *callback* could generate the elements?
- Such a callback could check Eddie to see if the requested `Image` was available, *load only that* (instead of everything into memory) and display it.
- This would be a solution to live/streaming data as well.
- This became the `DynamicMap`...

# DynamicMap

- `DynamicMap` is a subclass of `HoloMap` that uses a callback instead of a dictionary of values.
- Using `__getitem__` acted like indexing a `HoloMap` but instead the values were passed to the callback to generate the corresponding element.
- Initially a `DynamicMap` using Bokeh couldn't do much more than a `DynamicMap` using Matplotlib.
- That seemed like a shame as Bokeh had interactive tools to allow panning, zooming, selecting etc.
- Unfortunately, at the time Bokeh (written in coffeescript!) had no event system...

# An opportunity...

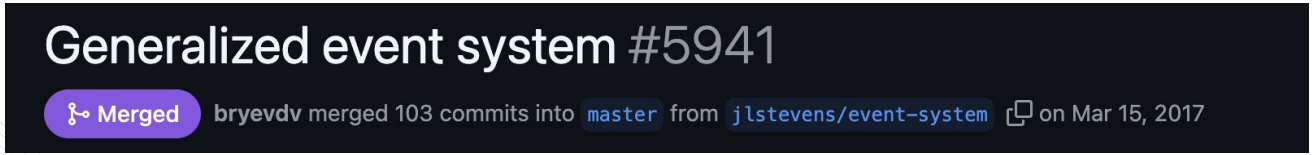
- Jim was using datashader to make lots of (static) pretty pictures
- Unfortunately, static images are not very useful for scientific exploration..
- Philipp and I could now be paid to work on open source (including HoloViews)
- We wanted to expose the interactivity of Bokeh through `DynamicMap`...
- ... but *unfortunately*, Bokeh lacked an event system...
- But *fortunately* Bokeh was another Continuum project...



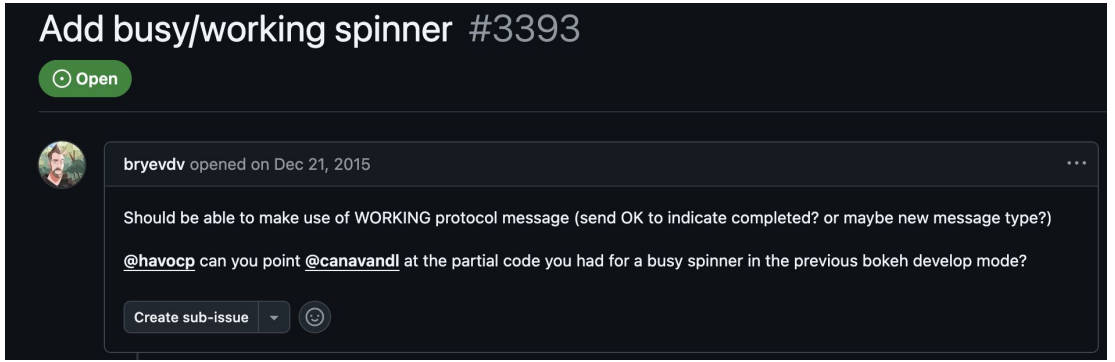
# Streams

# Bokeh events..

- Someone in 2015 opened issue #3393 to add a busy spinner to Bokeh.
- So in 2017, I used PR #5941 to solve it by giving Bokeh a general event system.
- This event system would allow Python to access view bounds and update on pan/zoom/clicks etc.
- Status of the PR:




- Status of the issue (2025):



# Modeling events in HoloViews

- Now Bokeh could send event information to Python (and therefore to HoloViews)
- As a source of live data, it was natural to use `DynamicMap` to display live updates.

**Problems:**

- How would HoloViews represent *all* events in a general way?
- How would all the types of interaction (tap, zoom, pan etc.) be represented?
- How would HoloViews represent Bokeh events in a way that could generalize across backends? (e.g. support plotly)
- How can you update a DynamicMap *that has already been displayed* (notebook)?
- Wouldn't it be nice to model event data with `param`?
- Should `param` be handling events at all?

# The need for `Streams`...

**Solution:**

- A parameterized `Stream` baseclass where parameters represent live data.
- The parameters of a stream then map to a `DynamicMap` callable by keyword.
- Instantiating a stream object would give you a handle with which to trigger an update through the plotting system.
- `DynamicMap` would then need a `streams` keyword to accept one or more streams.
- To make this work, data from ***all*** the stream objects would need to be collected before invoking the callback (i.e. to get a union of required keywords).
- Streams would need to use a defined API (e.g. `trigger`, `update` methods) to work. *Param didn't have an event system!*
- Each stream would have *subscribers* - callbacks that depend on that stream's value.

# Backend independent streams..

- Nothing in the design so far mentions the plotting backend:
  - You can use streams without any backend imported!
  - You can define arbitrary stream objects with the `Stream.define` classmethod.
  - Once you have a handle on the stream, you can trigger display updates (e.g. from another cell in a notebook session)
- Useful backend-independent streams include `Buffer` and `Pipe` (useful for streaming data applications).
- It is helpful to decouple the notion of streams from any specific backend, allowing streams to be expressed in user code without worrying about what is happening in the plotting backend.
- The correct way to view stream parameters is via the `.contents` property

# Linked streams

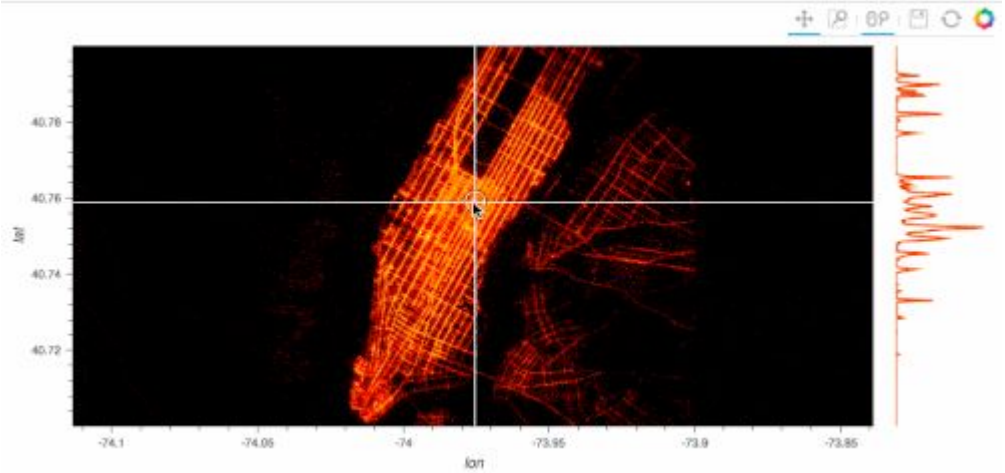
- Concepts like `Tap, PointerXY, PlotSize` don't make sense without something displayed by an interactive plotting backend.
- These are the *linked streams* and their implementation is tied to the plotting backend.
- However, the classes are defined in streams.py so that their semantics are defined independent of the backend library.
- For instance, Bokeh support is in `holoviews/plotting/bokeh/callbacks.py`
- This is where `Streams` are connected to Bokeh events, allowing Bokeh to drive the stream.
- When a `DynamicMap` is displayed, the plotting machinery registers callbacks with the stream subscribers so that updates from Bokeh drive updates to the display.

# `Callable`

- In order to validate a `DynamicMap` callable, it is necessary to knows its keywords.
- This is one of the reasons a `DynamicMap` callable is wrapped into a `Callable` proxy.
- This also allows caching (memoization) to avoid re-invoking a callback for a previously seen set of arguments.
- It also supports Python generators (a feature rarely used nowadays)
- `DynamicMap.cache_size` can be set to zero to disable this caching (e.g. for stateful callbacks).
- `Callable` is also where stream parameters are collected and supplied to the user-supplied callback.

# Interactive datashader

- Datashader was able to generate static images, really, really fast using numba.
- By adding datashader support to HoloViews, streams could finally allow datashader to be used interactively*
- This was the first really compelling use of streams that worked ‘out of the box’




* there was another approach we won’t mention!

# Outcomes

- Streams have become the basis of all the more advanced interactivity offered by HoloViews.
- Interactive datashader is now easily available via `hvplot`.
- Examples include holodoodler, holonote, timeseries explorer...
- Datashader-like interactivity now extends beyond `Images` (e.g. LTTB)
- New work integrates Bokeh popups into Streams via the `popup` keyword.
- Bokeh is now the primary backend for HoloViews (the matplotlib backend is languishing a little).

# Things move on…

- HoloViews was the first part of the HoloViz ecosystem to gain an event system.
- A lot of what was learned informed the design of panel and the param events system.
- Param now has many, many, many ways to express events.
- The `Params` stream acts as a bridge between param events and HoloViews streams.
- Having the `streams` argument of `DynamicMap` be a list proved a little too restrictive: you can now pass in a dictionary to map parameters or panel widgets to keywords.

# Thank you for listening!

# Any Questions?