Plotting large datasets, Part 3: Advanced experiments

Colin Leach, March 2020

This should really be called "Things I don't understand yet". It won't be much use for 400B work, but I've posted it for the tiny minority who might be interested.

Getting away from matplotlib, there are some newer[1] options which use the power of modern graphics cards to handle large numbers of points. More powerful, steeper learning curve.

1 datashader

Websites: `https://github.com/holoviz/datashader` and `https://datashader.org`

This needs the data to be arranged columnwise in a (single) pandas dataframe; fairly typical for modern plotting packages. Used in isolation, datashader will make a 2D image. To add axes, labels, etc, it needs to be hooked into a suitable browser-based package such as Bokeh (usually) or Plotly (under development). This can mean a LOT of extra packages to install from conda or pip and lots of potential conflicts[2]

It was a fight for me to get something working. Documentation is extensive, but sometimes out of date and actively misleading (it took over an hour to change the colormap at about the 20th attempt). The best result so far starts with a bunch of imports:

```
import pandas as pd
import numpy as np
from matplotlib import cm # use the same colormap as other examples

import holoviews as hv
import holoviews.operation.datashader as hd
hv.extension("bokeh")
hv.output(backend="bokeh")

import datashader as ds
```

Then massage the data into the right format and plot it:

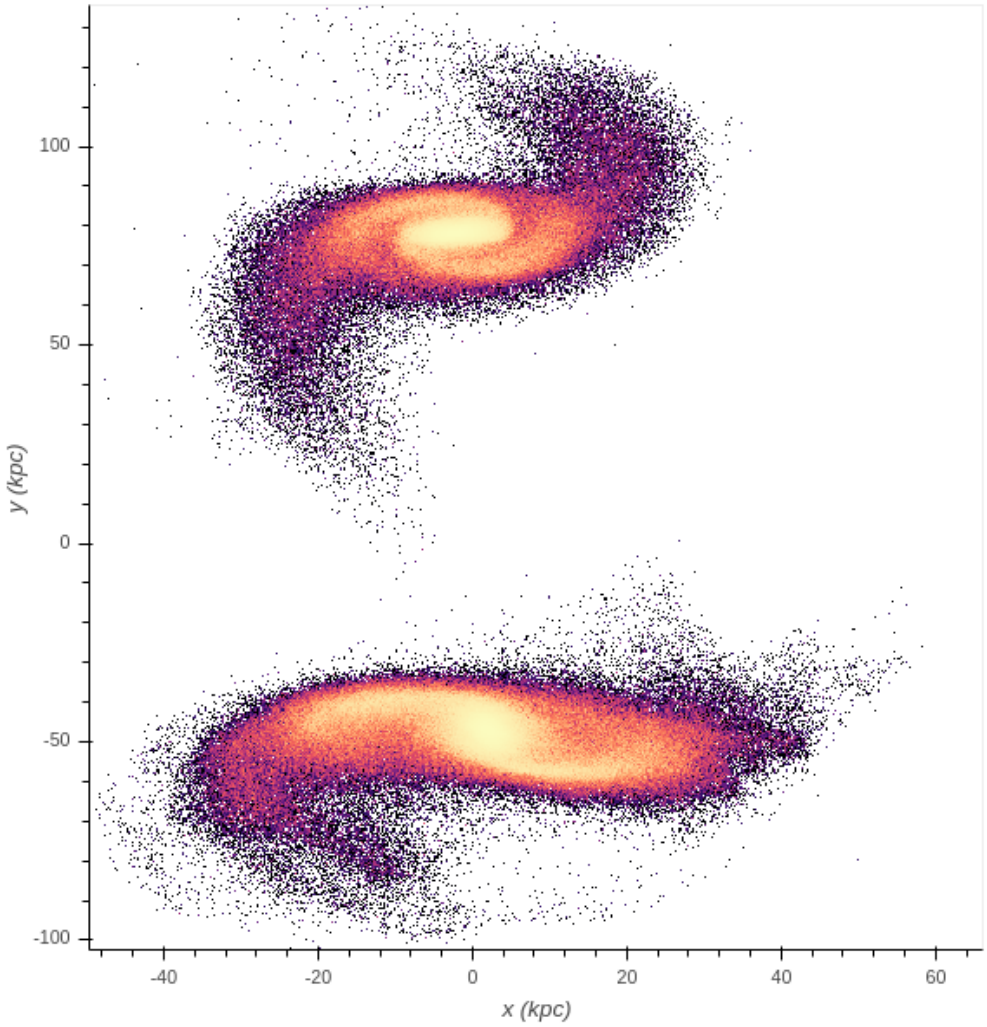
```
# convert np.array to pandas dataframe
points_df = pd.DataFrame(disks.T, columns=('x (kpc)','y (kpc)','z (kpc)'))

# convert df to Holoviews points
points = hv.Points(points_df)

# plot in Bokeh
hd.datashade(points, cmap=cm.magma).opts(height=600, width=600)
```

[1] In other words, typically less finished and poorly documented

[2] The recent release of Bokeh 2.0 broke datashader 0.10.0. For now, use Bokeh 1.4.0 and hope datashader 0.13 is released soon. The developer says he's working on it while in self-isolation after flying home from Covid-hit Spain.

The results aren't terrible (see below), but we've still to make the cmap logarithmic, fix the font sizes and add a title. All possible, but perhaps another day...

Meanwhile, the best feature is it lets you (in Jupyter) pan with mouse drags and zoom with the mouse wheel or box selection. Matplotlib can't do that so easily (though the "%matplotlib notebook" IPython magic is an option worth knowing about).



2 ipyvolume

Works natively with 3D data. Only in the browser as it relies on WebGL rendering and Javascript.

Websites: `https://github.com/maartenbreddels/ipyvolume` and `https://ipyvolume.readthedocs.io/en/latest/`

Spectacular when the developer (Maarten Breddels, a Dutch astronomer) demonstrates it at conferences, not so easy for the rest of us. I totally failed with this about a year ago, still psyching myself up for another attempt. Anyone else tried it?

3 All things OpenGL

Now we're at the heavy end of the options. With pyqt5 + (pyopengl or vispy), an experienced graphics programmer with unlimited time can do virtually anything. Though the same person would probably ignore Python and use C++ with GLSL; probably wise, because the C++ libraries at least have excellent documentation.

I played with this a bit over spring break. At the current rate of progress, I may have a crude prototype working by the end of the Spring 2021 semester (but no promises).