Meaningful Visual Exploration of Massive Data

Peter Wang
Continuum Analytics
CTO & Co-Founder

@pwang
Big data magnifies small problems

- Big data presents storage and computation problems
- More importantly, standard plotting tools have problems that are magnified by big data:
  - Overdrawing/Overplotting
  - Saturation
  - Undersaturation
  - Binning issues
Overdrawing

- For a scatterplot, the order in which points are drawn is very important.
- The same distribution can look entirely different depending on plotting order.
- Last data plotted overplots.
Overdrawing

- Underlying issue is just occlusion
- Same problem happens with one category, but less obvious
- Can prevent occlusion using transparency
Saturation

- E.g. for alpha = 0.1, up to 10 points can overlap before saturating the available brightness

- Now the order of plotting matters less

- After 10 points, first-plotted data still lost

- For one category, 10, 20, or 2000 points overlapping will look identical
Saturation

- Same alpha value, more points:
- Now is highly misleading
- Alpha value depends on size, overlap of dataset
- Difficult-to-set parameter, hard to know when data is misrepresented
Saturation

- Can try to reduce point size to reduce overplotting and saturation
- Now points are hard to see, with no guarantee of avoiding problems
- Another difficult-to-set parameter

- For really big data, scatterplots start to become very inefficient, because there are many datapoints per pixel — may as well be binning by pixel
Binning issues

- Can use heatmap instead of scatter
- Avoids saturation by auto-ranging on bins
- Result independent of data size
- Here two merged normal distributions look very different at different binning
- Another difficult-to-set parameter
Notebook

Occlusion of data by other data is called overplotting or overdrawing, and it occurs whenever a datapoint or curve is plotted on top of another datapoint or curve. This is thus a problem not just for such points as these, but for pure points, as well.
Real world example
So what?
Plotting big data

- When exploring really big data, the visualization is all you have — there’s no way to look at each of the individual data points.
- Common plotting problems can lead to completely incorrect conclusions based on misleading visualizations.
- Slow processing makes trial and error approach ineffective.

When data is large, you don’t know when the viz is lying.
Datashading

• Flexible, configurable pipeline for automatic plotting
• Provides flexible plugins for viz stages, like in graphics shaders
• Completely prevents overplotting, saturation, and undersaturation
• Mitigates binning issues by providing fully interactive exploration in web browsers, even of very large datasets on ordinary machines
• Statistical transformations of data are a first-class aspect of the visualization
• Allows rapid iteration of visual styles & configs, interactive selections and filtering, to support data exploration
Datashading Pipeline: Projection

- Stage 1: select variables (columns) to project onto the screen
- Data often filtered at this stage
Datashading Pipeline: Aggregation

- Stage 2: Aggregate data into a fixed set of bins
- Each bin yields one or more scalars (total count, mean, stddev, etc.)
Data

Stage 3: Transform data using one or more transfer functions, culminating in a function that yields a visible image.

Each stage can be replaced and configured separately.
Infovis Pipeline

Source Data → Data Tables → Visual Abstraction → Views

Data Transforms, Visual Mappings, View Transforms
Infovis Pipeline… modified
NYC Taxi Demo

This demo shows how traditional plotting tools break down for large datasets, and how to use datashading to make even large datasets practical interactively.

- Data for 10 million New York City taxi trips
- Even 100,000 points gets slow for scatterplot
- Parameters usually need adjusting for every zoom
- True relationships within data not visible in std plot

Datashading automatically reveals the entire dataset, including outliers, hot spots, and missing data.
Categorical data: 2010 US Census

- One point per person
- 300 million total
- Categorized by race
- Datashading shows faithful distribution per pixel
OSM Dataset: 3 Billion Points

Because Datashader decouples the data-processing from the visualization, it can handle arbitrarily large data

- About 3 billion GPS coordinates
- [https://blog.openstreetmap.org/2012/04/01/bulk-gps-point-data/](https://blog.openstreetmap.org/2012/04/01/bulk-gps-point-data/)
- This image was rendered in one minute on a standard MacBook with 16 GB RAM
- Renders in 7 seconds on a 128GB Amazon EC2 instance
Additional

- Timeseries
- Trajectory
- Race & Elevation
- API / Pipeline walkthrough
Bokeh

- Interactive visualization
- Novel graphics
- Streaming, dynamic, large data
- For the browser, with or without a server
- No need to write Javascript

http://bokeh.pydata.org
Versatile Plotting Capabilities
Linked plots, tools

• Easy to show multiple plots and link them
• Easy to link data selections between plots
• Can easily customize the kind of linkage straight from Python, without needing to fiddle around with JS
Large data

- With easy WebGL support, can scale to 500k points or so
- Bottlenecks are browser performance, JSON encoding, network transport
rBokeh

Plays well with R ecosystem: HTMLwidget, RMarkdown…

http://hafen.github.io/rbokeh
rBokeh with RStudio & Shiny

Plays well with R ecosystem: HTMLwidget, RMarkdown…
Bokeh Apps: Shiny for Python

- Fully interactive data web apps
- Streaming data, dynamic data
- Easy-to-write pure Python charts, widgets, event handlers
- Open source (BSD licensed), including server
- *Enterprise on-prem version in Anaconda Enterprise, with Active Directory/LDAP auth*
Example Apps
Easy Streaming Apps

In this demo, we will demonstrate how the Bokeh server makes it easy to visualize streaming and dynamic data.

• A minimal example with < 50 LOC
• Demonstrates ease of pushing data from Python code into the browser
Embeds Well

http://cecp.mit.edu
For more information on Bokeh Apps

• Webinar: http://www.slideshare.net/continuumio/hassle-free-data-science-apps-with-bokeh-webinar

• PyData Videos, Tutorials
Community & Adoption

Github
• 4100+ stars
• 860+ forks

Mailing list
• 400+ members
• 150+ posts in November

Downloads
• 45,000 / month (conda)
• 4,000 / month (pip)
Architecture
Traditional Web Visualization

Tech:
- Python/R/Java
- HTML & browser compat
- CSS/LESS/Sass
- JS plotting library API
- Javascript
  - jQuery, underscore
  - svg, canvas2D
  - WebGL, three.js
  - React
  - Angular
  - node.js, browserify, gulp, grunt, npm, …
Simple dashboard: Server language generating HTML, JS, CSS styling, subset of data

Handling user interaction: Custom Javascript, calling Server endpoint, which generates updated JSON or JS that gets pushed back to client via websocket
Bokeh Conceptual Architecture

Simple dashboard: Single language, no need to write HTML, JS, CSS

Handling user interaction: Single language that you already know; interactive data updates feel seamless to the user
Comparison Chart

**Server**
- Python, Ruby
- Java, .NET

**Client**
- Python, R
- Bokeh

**Client**
- BokehJS

- Skills required: 5-10 skills
- Time to market: weeks to months
- Server code: 100s to 1000s lines

- Skills required: ~1 skill
- Time to market: minutes
- Server code: 0
Thank you

Peter Wang
Twitter: @pwang

Bokeh
Twitter: @bokehplots

Email: sales@continuum.io
Twitter: @ContinuumIO