Interactive Visualization

Sweden's vast Bird Fauna

Nils Fahrni
BSc Data Science Student

November 25, 2024

Contents

Chapter 1
Performance

When working with interactive visualizations, performance plays a crucial role in ensuring a smooth user experience, especially as datasets grow larger and more complex. With increasing data points and variables, the demand for both computational power and efficient rendering becomes essential. Interactive visualizations can encounter performance bottlenecks due to factors such as data size, rendering techniques, bandwidth, and hardware limitations. As data size increases, performance issues can manifest in the form of delayed rendering, increased memory usage, and ultimately, a sluggish or unresponsive interface. These challenges make it essential to understand and implement performance-enhancing solutions such as tiling, level of detail (LOD) management, GPU acceleration, and hierarchical data organization.

In this context, two experiments were conducted to benchmark and understand the performance implications of interactive visualizations, both in terms of data size and choice of visualization libraries. The following sections analyze these experiments, where Figure **??** and Figure **??** showcase the performance metrics obtained.

1.1 Experiment 1: Impact of Data Size

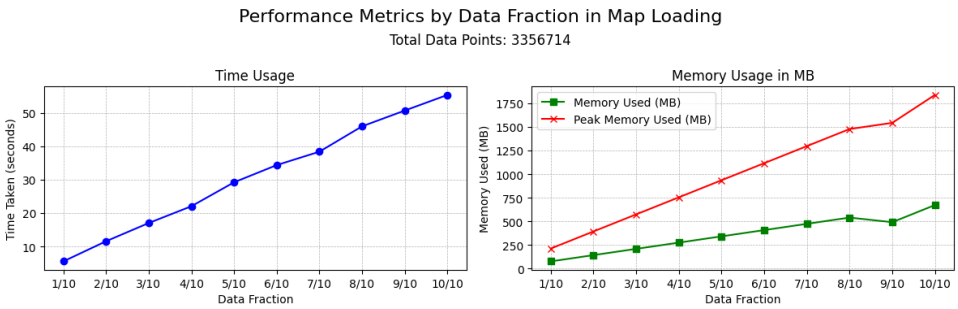

Figure 1.1: Performance metrics for incremental data loading in a map visualization.

Figure **??** illustrates the performance metrics captured when incrementally loading fractions of a dataset into a map visualization. In this experiment, each data point represented a bird observation in Sweden, and fractions of the dataset were loaded from 1/10th up to the entire dataset (10/10ths, or 3'356'714 observations). Two metrics were recorded: Time Usage (in seconds) and Memory Usage (in MB).

1.1.1 Time Usage Analysis

The left subplot of **??** shows a linear trend in time consumption as the data fraction increases. The time taken rises from approximately 10 seconds for 1/10th of the data to over 50 seconds for the entire dataset. This trend reflects the computational load associated with managing and rendering larger datasets in real-time.

This increase in time usage can be attributed to the overhead of both loading data into memory and the rendering process itself. With each increment, the visualization needs to plot more points on the map, which places a growing strain on both the CPU and GPU, depending on the level of interactivity and graphical requirements.

1.1.2 Memory Usage Analysis

The right panel of Figure 1 captures memory usage with two lines: one for Memory Used and another for Peak Memory Used. Memory Used remains relatively lower, while Peak Memory steadily climbs with

each data increment.

The growing gap between regular and peak memory usage can be attributed to the temporary memory required to process the data before rendering. As the data size increases, the visualization needs to allocate more temporary memory for computation, even if the plotted memory stabilizes.

This memory trend suggests that managing data efficiently, such as by tiling or downsampling, could reduce the memory load, making the visualization more responsive without compromising on detail.

1.2 Experiment 2: Impact of Plotting Library on Performance

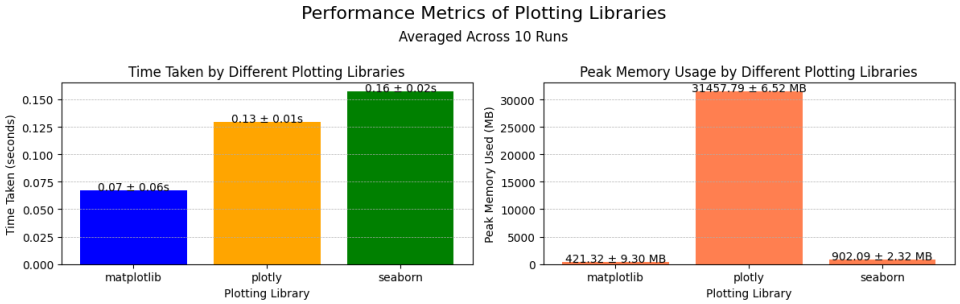

Figure 1.2: Performance metrics for different plotting libraries.

In Figure **??**, the performance of three plotting libraries—Matplotlib, Plotly, and Seaborn—was measured by plotting the frequency of the 10 most common bird species in Sweden. To ensure accuracy and confidence in the results, each test was run 10 times, and the Time Taken and Peak Memory Usage metrics were averaged across these runs, with standard deviations included to quantify variability.

1.2.1 Time Taken Analysis

The left panel of Figure **??** shows that Matplotlib remains the fastest library, with an average time of 0.07 seconds ± 0.06 seconds. Plotly and Seaborn, however, took considerably longer, with average times of 0.14 seconds ± 0.02 seconds for Plotly and 0.15 seconds ± 0.01 seconds for Seaborn.

The high standard deviation for Matplotlib indicates some variability in its runtime, possibly due to fluctuations in processing lower-level operations. In contrast, Plotly and Seaborn show relatively stable performance with smaller standard deviations, reflecting their structured approach to rendering but at the cost of increased processing time.

Matplotlib's efficiency is likely due to its streamlined focus on static plotting, while Plotly's interactivity and Seaborn's additional styling complexity add to their time overhead. These findings emphasize that while Matplotlib is optimal for static visualizations, Plotly and Seaborn introduce extra time due to their functionality and aesthetic layers, respectively.

1.2.2 Memory Usage Analysis

The right panel of Figure **??** reveals a stark contrast in peak memory usage among the libraries. Plotly uses significantly more memory, averaging 31,471.73 MB ± 40.09 MB, highlighting the high cost of interactivity. Seaborn requires considerably less memory, averaging 903.09 MB ± 2.13 MB, and Matplotlib remains the most memory-efficient, with an average of 422.25 MB ± 9.62 MB.

The consistent but high memory usage for Plotly suggests that its interactive features, such as hover and zoom functionality, impose a substantial memory footprint. These features, while enhancing user engagement, demand extensive resources for rendering and data management, particularly when handling large datasets or complex visualizations.

Seaborn, despite being built on top of Matplotlib, requires additional memory to manage stylistic elements. Matplotlib's low memory usage reaffirms its suitability for static visualizations where memory constraints are critical. This suggests that when designing interactive visualizations with large datasets,

it is essential to account for Plotly's memory requirements, while Matplotlib and Seaborn serve as better choices for simpler plots.

Chapter 2
Design Principles

2.1 Introduction to Shneiderman's Mantra

In the realm of information visualization, effective data exploration is paramount. Ben Shneiderman, a prominent figure in human-computer interaction, proposed a foundational guideline known as Shneiderman's Mantra to enhance user interaction with complex datasets. The mantra succinctly states: *"Overview first, zoom and filter, then details-on-demand"* **hampdatavisualizationSchneidermansMantra2016**. This principle serves as a blueprint for designing intuitive interfaces that facilitate efficient data analysis.

Shneiderman's Mantra emphasizes a hierarchical approach to data exploration:

1. **Overview First**: Present the entire dataset in a high-level view to provide context and scope.
2. **Zoom and Filter**: Allow users to focus on subsets of the data that interest them.
3. **Details on Demand**: Enable access to detailed information about specific data points when required.

By adhering to this sequence, interfaces can cater to both novice users and experts, providing a scalable means to interact with data of varying complexity.

2.2 Implementation in the Birds of Sweden Dashboard

The Birds of Sweden Dashboard was developed to enable users to navigate and explore Sweden's bird fauna interactively. Implementing Shneiderman's Mantra within this dashboard ensures that users can effectively analyze bird observation data across the country.

2.2.1 Overview First

Upon launching the dashboard, users are presented with a comprehensive map displaying all bird observations across Sweden (Figure **??**). This initial view provides an immediate sense of the spatial distribution and density of bird sightings, fulfilling the "Overview First" principle.

The map utilizes a scatter plot where each point represents an observation, allowing users to perceive patterns such as hotspots of biodiversity or migratory pathways. Additionally, the sidebar includes a state observations map that consistently displays the entire country's outline, highlighting the number of observations per state. This reinforces the comprehensive nature of the overview.

2.2.2 Zoom and Filter

To facilitate more focused exploration, the dashboard incorporates interactive filtering mechanisms. A dropdown menu enables users to select a specific bird species from an extensive list derived from the dataset (Figure **??**). Upon selection, the main map updates to display only the observations pertaining to the chosen species.

This filtering capability exemplifies the "Zoom and Filter" principle, allowing users to narrow down the dataset to areas or species of interest. The state observations map in the sidebar also updates accordingly, highlighting only the states where the selected species has been observed, thus providing a filtered geographical context (Figure **??**).

2.2.3 Details on Demand

To offer in-depth information, the dashboard provides detailed data about specific observations and species. When a species is selected, the sidebar displays:

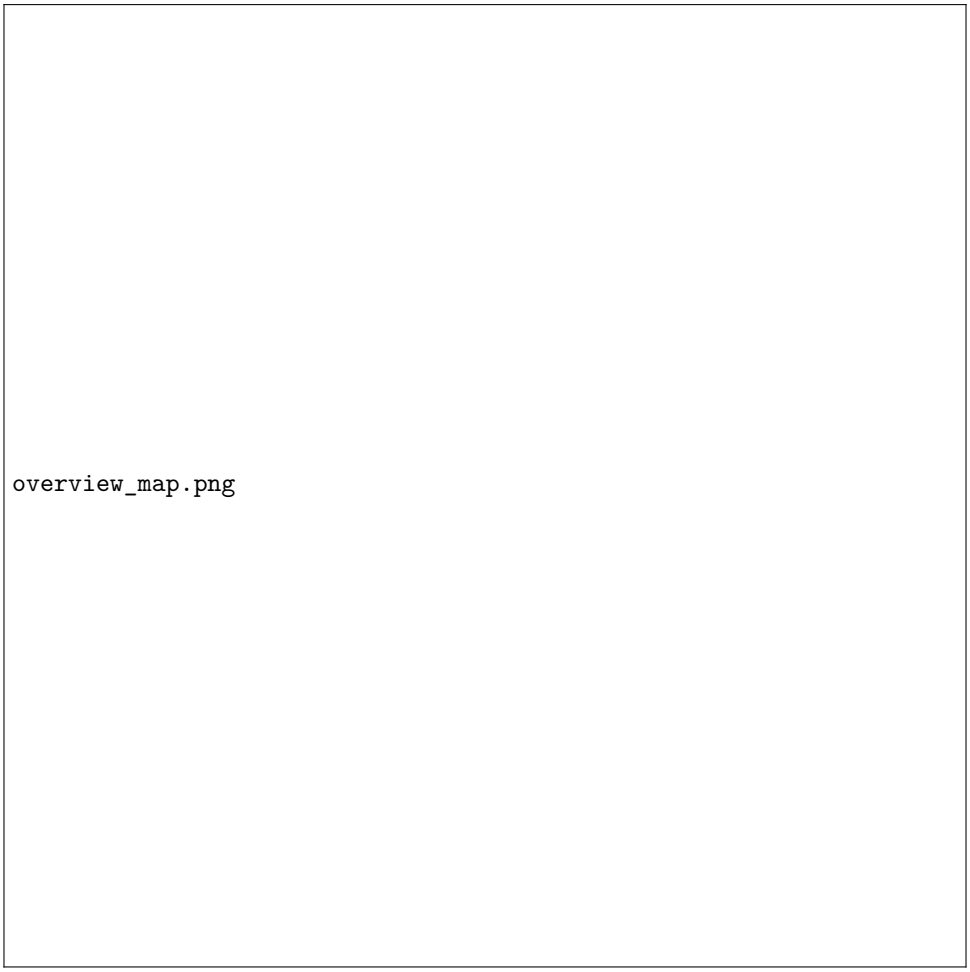

Figure 2.1: Initial overview of all bird observations in Sweden.

- **Species Name and Scientific Name**: Clarifies the common and scientific nomenclature.
- **Species Image**: Retrieves an image from Wikipedia for visual identification.
- **Additional Information**: Presents taxonomic order, observation count, breeding category, state, and locality.

This implementation of "Details on Demand" ensures that users can access specific information without overwhelming the initial view. Moreover, clicking on an observation point on the map sets the species in the dropdown, updating the sidebar with relevant details (Figure **??**).

2.2.4 Technical Implementation

The dashboard leverages the Dash framework for Python, enabling the creation of interactive web applications. Key implementation aspects include:

- **Data Loading**: The full dataset is loaded without slicing to ensure all observations and species are available.
- **Data Cleaning**: Missing values in critical columns such as `LATITUDE`, `LONGITUDE`, and `COMMON NAME` are handled to prevent empty maps or dropdowns.
- **Callbacks**: Dash callbacks are used to update the map and sidebar components dynamically based on user interactions.
- **Geographical Consistency**: The state observations map uses the `fitbounds="geojson"` parameter to always display the entire country's outline, regardless of the data filtered.

Figure 2.2: Species selection dropdown for filtering observations.

By addressing initial issues such as empty maps on startup and mismatches in state naming conventions, the dashboard now aligns with the intended functionality and provides a seamless user experience.

2.3 Goals and Hopes for the Dashboard

The primary objective of implementing Shneiderman's Mantra in the Birds of Sweden Dashboard is to enhance user interaction and facilitate the exploration of bird fauna data. The specific goals include:

2.3.1 Improving User Interaction and Experience

By providing an intuitive interface that adheres to established principles of information visualization, users can navigate the dataset effectively, regardless of their familiarity with the data. The hierarchical approach reduces cognitive load and allows for a more engaging experience.

2.3.2 Facilitating Exploration of Sweden's Bird Fauna

The dashboard serves as an educational tool, enabling users to discover patterns and insights within the bird observation data. For instance, users can identify regions with high biodiversity, track migration patterns, or explore species distribution.

2.3.3 Enabling Users to Gain Insights

Through interactive filtering and detailed information access, users can perform analyses that may lead to new findings or support research efforts. Conservationists, ornithologists, and bird enthusiasts alike can leverage the dashboard to inform their work or interests.

2.4 Conclusion

Implementing Shneiderman's Mantra in the Birds of Sweden Dashboard has resulted in a powerful tool for data exploration. By providing an initial overview, enabling focused filtering, and offering detailed information on demand, the dashboard aligns with best practices in information visualization. The hope is that this approach not only enhances user engagement but also contributes to a deeper understanding of Sweden's rich bird fauna.

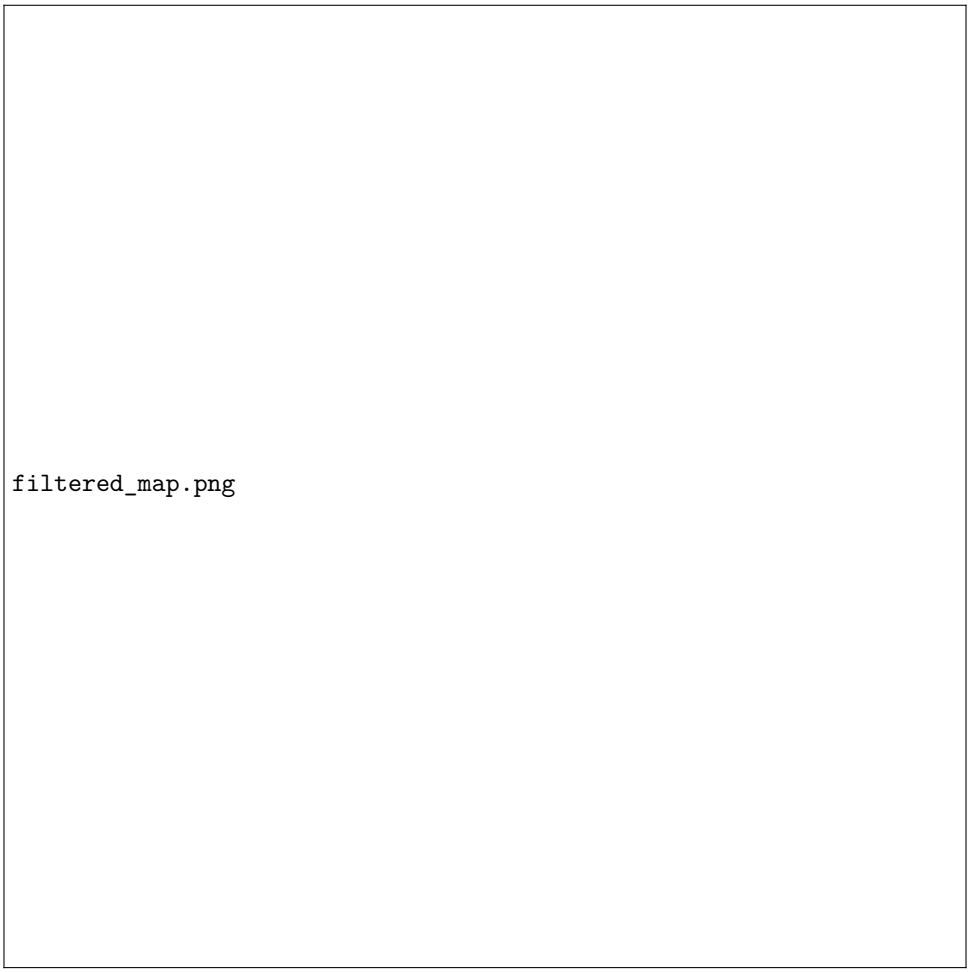

Figure 2.3: Map displaying observations of a selected species after filtering.

Figure 2.4: Detailed information displayed upon species selection.

Chapter 3
HCI Basics

Chapter 4
Evaluation