Sherpa: Leveraging User Attention for Computational Steering in Visual Analytics

Zhe Cui, Jayaram Kancherla, Héctor Corrada Bravo, and Niklas Elmqvist
University of Maryland, College Park

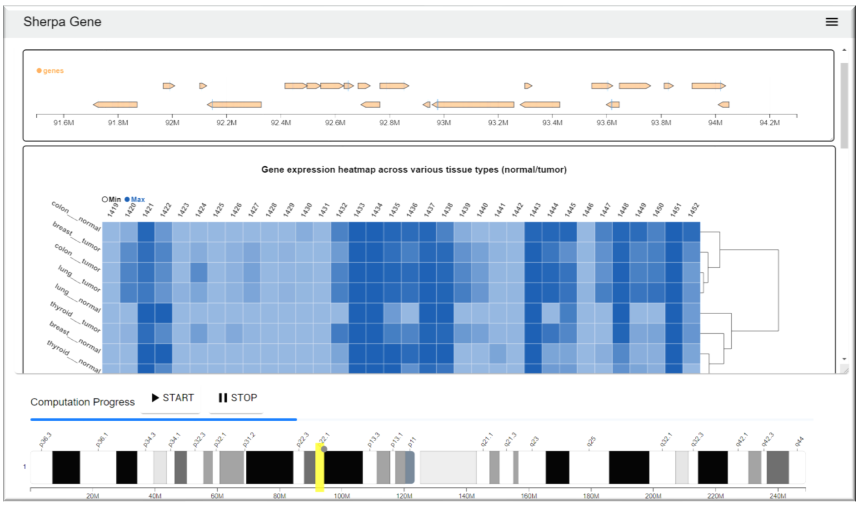

Figure 1: Sherpa Gene: a web-based visual analytics application for genomics incorporating attention-based computational steering. The gene track and gene expression heatmap display the user's current focus. The ideogram (bottom) serves as the data space view on which the user controls the yellow navigation window, which governs computational priority.

Abstract

We present Sherpa, a computational steering mechanism for progressive visual analytics that automatically prioritizes computations based on the analyst's navigational behavior in the data. The intuition is that navigation in data space is an indication of the analyst's interest in the data. Sherpa implementation provides computational modules, such as statistics of biological inferences about gene regulation. The position of the navigation window on the genomic sequence over time is used to prioritize computations. In a study with genomic and visualization analysts, we found that Sherpa provided comparable accuracy to the offline condition, where computations were completed prior to analysis, with shorter completion times. We also provide a second example on stock market analysis.

1 Introduction

As per the definition of visual analytics (VA) [35], many VA applications require significant computations—such as clustering [19], word embedding [14], and inferential statistics—to be run on new datasets prior to presentation to the user. However, real-world datasets are increasingly reaching a volume and complexity where such computation can be forbiddingly costly in terms of time and resource which the analyst may not be able to spare. To remedy this, big data analytics [12] is increasingly turning to partial, progressive, and incremental methods, where instead of waiting for computation to finish prior to viewing the data, the user is shown an intermediate view of the data that is continuously updated throughout the computation [1, 47]. While advanced database techniques can provide reliable partial results of even large datasets [13], we could be using our computational resources more efficiently if we knew which part of the data the user was interested in. For example, given ten years of stock market data, clustering stock trends for each time segment starting from the beginning of the recorded time period is inefficient if the user is only interested in the stock from last year. Unfortunately, to ensure efficient usage of the tool, most current interfaces for this kind of *computational steering* [27, 43] of time-consuming algorithms often require the user to have significant expertise of the computation itself, which only few data analysts possess.

We propose SHERPA, a method for leveraging the user's attention to implicitly derive priorities for computational operations on a dataset. The idea is straightforward: Sherpa provides a *data space view* (Figure 1) where the user can control their current locus of attention using a *navigation window*. For example, in the 10-year stock market example, the user may pan and zoom their navigation window to focus solely on the last year in the dataset. Sherpa uses the dynamically changing navigation window to implicitly derive the priority of computation for each portion of the dataset. The Sherpa scheduler will prioritize finishing calculations for those areas of the dataset that the user has expressed an interest in using the navigation window. The main visualization will show a progressively updating view of the currently selected subset of the data as computation proceeds. Priorities decay over time, allowing the user to change their focus throughout an analysis process.

The Sherpa method is independent of application, and could be applied to any dataset provided that computations can be localized to specific regions of the data, such as the stock market, time-series data, and local network metrics. We have implemented Sherpa in two application domains: genomics and stock market. In the human genomics application, where multiple data modalities—gene expression and DNA methylation—across four cell types (colon, lung, breast and thyroid) and their corresponding normal tissue are spatially indexed over genomic position across 23 chromosomes (over

3 billion possible positions in total). The computations in this application are statistical inferences that reveal mechanisms underlying gene regulation and expression. The progressively updated visualization (Figure 1) shows a track displaying gene location and structure within the focused genomic region, another track displaying genomic blocks of significant methylation difference between tumor and normal tissue, and a heatmap of gene expression across multiple tissues. The tool will gradually add multiple small scatterplots and block tracks to the main visualization view for statistically significant tests (based on correlation and block overlap computations). The data space view uses the spatial position within the chromosome to order data, and the user's movement of the navigation window will change the priorities of which data to compute.

We have evaluated our Sherpa implementation for the genomics application under three conditions: (1) a *classic static* condition, where only the final computation is shown to the user (which serves as a baseline); (2) a *progressively updating* condition, where the display updates as computation proceeds but where the user cannot steer the computation; and (3) a *Sherpa* condition, where the user's navigational behavior on the sequence will steer the order of computation. In our study, participants were asked to answer high-level questions about specific aspects of the data. Our results show that implicit computational steering using the Sherpa approach provides significant time improvements for tasks that are specific to known gene locations (e.g., specific genes of interest). This suggests that computational steering can be beneficial for visual analytics, even when the user lacks the expertise to control the computation.

We propose the following contributions: (1) the Sherpa method for implicitly deriving computational priorities for a dataset based on the user's navigational behavior in data space as a proxy of their attention; (2) a Sherpa implementation for genomic data (Sherpa Gene) for multiple tracks over an entire chromosome sequence, where the computation calculates test statistics based on genomic region overlaps and correlation of quantitative measurements indexed by genomic location, as well as one for stock market data (Sherpa Stock) for calculating market indicators at scale for a massive financial dataset; and (3) results from a user study comparing implicit computational steering using Sherpa with two baseline conditions for the genomics application.

2 Background

Our work straddles several topics: computational steering, progressive visual analytics, mixed-initiative interaction, and visit wear.

2.1 Computational Steering

Many computational algorithms, particularly for scientific and simulation purposes, are extremely resource-intensive and time-consuming, often requiring massive computational clusters. For this reason, the notion of *computational steering* —interactive control over a computation during execution [39]— is very attractive, as it allows the scientist or engineer to guide the process in real-time in order to faster converge on a desirable solution. Mulder et al. [27] enumerate uses of computational steering as model exploration, algorithm experimentation, and performance optimization. Examples of well-known computational steering environments include SCIRun [28], Progress/Magellan [41, 42], and VASE [18]; some applications include fluid dynamics (CFD) [7], program and resource steering [41], and high performance computing (HPC) platforms [4].

Most computational steering mechanisms are *explicit*, in that they give the user control over the ongoing computation using operations that are specific to the domain. However, this requires significant expertise on behalf of the user. Recent efforts have focused on coupling interactive visualization with computational steering to display intermediate results as well as provide direct controls. World Lines [44], Nodes on Ropes [45], and Visdom [32] are all examples of such integrated steering environments, typically used to control multiple runs of the same or related simulation models with slightly perturbed inputs. Similarly, VASA [23] is a VA system for asynchronous computational steering of large simulation pipelines.

2.2 Progressive Visual Analytics

The tremendous leap in computational power over the last few decades has so far mostly benefited confirmatory analysis, where the analyst initializes a model and executes it, waiting minutes, hours, and sometimes days for the computation to finish. A more exploratory data analysis [38], such as those supported by interactive visualization and analytics [35], requires a tightly optimized feedback loop with latency of at most 10 seconds (often around 0.5 seconds [25]). To make big data analytics [12] responsive in such interactive and exploratory settings, recent work has proposed the concept of *progressive visual analytics* (PVA) [1, 47], where intermediate results are continuously fed back to the visualization to show gradual progress. While PVA nominally includes computational steering as one of its main components [47], few practical implementations provide steering capabilities. The original ProgressiVis Python toolkit [11] has "optional input slots", but these are never explained in detail. Zgraggen et al. [47] evaluate PVA for three output conditions, including blocking, instantaneous, and progressive, but do not involve user-controlled steering in their study. PANENE [20] is a progressive tree structure for nearest neighbor computations, but does not expose steering controls to users.

In contrast, Badam et al. [1] explore user interfaces for PVA in particular, providing process controls (pause, stop, and progress bars) and algorithm-specific options for controlling the ongoing execution. However, the process controls are simplistic, whereas the algorithm options merely expose the raw parameters of the computation, and thus require some expertise to manipulate. The incremental query visualizations proposed by Fisher et al. [13] provide similar basic controls for pausing, resuming, and canceling an ongoing query. Finally, a recent progressive implementation of t-SNE dimensionality reduction allows the user to control which part of the data to focus on first [29]. This approach is highly relevant to our approach in that it provides a user-controlled Magic Lens [5] that will also steer the computation. However, the approach is specific to t-SNE, and does not focus on the navigational behavior as a proxy for attention.

2.3 Mixed-Initiative Interaction

Computers are typically treated as tools, albeit highly advanced ones, but new trends are proposing a computer-as-partner paradigm [2, 3], where the digital medium itself is an active participant. One example of this paradigm is *mixed-initiative interaction*, where people and computers alike contribute to solving a task together [16, 17]. Incidentally, this type of dialog between man and machine is a canonical form of visual analytics [35], where a human's analytical process is aided by computational processes and visual interfaces.

Several examples of mixed-initiative interaction applied to visualization exist. Endert et al. [8, 9] propose semantic interaction, where the user's interaction is back-propagated into statistical models, allowing them to learn and change based on the user's implicit input. Finally, the VASA tool [23] lets the analyst explore "what-if" scenarios by interleaving their decision-making with computer simulations of weather, supply chains, and roads.

2.4 Visit Wear and View Mining

Examples of visit wear both include navigation histories [34], such as for web browsers, as well as the geographic footprints on online maps suggested by TrailMap [48] and GroupTrail [46].

What if we could determine user interest merely by what part of a digital object they view? *View mining* uses machine learning methods to extract common viewpoints based on user interaction data. Singh and Balakrishnan [33] mine camera movements in 3D space to extract optimal viewpoints for a scene. Most relevant to

Sherpa, Lagun and Lalmas [24] and Grusky et al. [14] study the concept of "viewport time"—the position of a user's viewport on a document over time—as an indication of user attention. In Grusky's study, navigation behavior was strongly correlated with eye-tracking data, yet requires no specific and costly hardware to capture.

3 Attention for Computational Steering

The Sherpa model is an implicit form of computational steering for priority-based processing of a dataset based on user's attention. The intuition behind the model is to prioritize computations on those areas of the dataset that the user deems important. We derive user attention from the location and dimensions of an interactive *navigation window* on an overview representation of the dataset (*data space view*). Figure 2 gives an overview of the Sherpa interface.

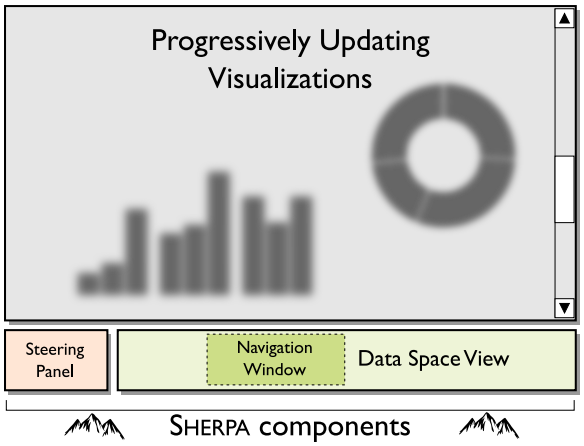

Figure 2: Overview of the Sherpa user interface components.

3.1 Basic Model

Sherpa is a general model that can be instantiated for specific applications, datasets, and computations. It makes a few assumptions about the application that we outline below.

First of all, Sherpa requires a *dataset* with natural location-based semantics; this could either be truly spatial, such as for locations on a map or positions in a gene sequence, or temporal, such as positions in time. The dataset should be of sufficient size where it cannot just be trivially processed prior to shown to the user; in such a case, the Sherpa method (or any other PVA method) is not necessary.

Second, Sherpa requires a corresponding *computation* on the dataset that can be performed on data items in random order. In other words, the computation must be *parallelizable* so that a particular computation has no dependencies to results for other parts of the data. Note that it is possible that computations could be restricted to chunks of items—such as genes in a genome—instead of individual items, as long as there are sufficiently many chunks so that their order of computation is significant.

Third, the method requires a *visualization* that (1) can represent a specific subset of the dataset, and (2) can be progressively updated over time as new calculations are completed. The former property is required so that the user's navigation in data space actually affects the main visualization view (otherwise there is no purpose of the user to navigate in the data space view); the latter is also needed so that the view can be refreshed as new results are produced.

3.2 Steering Functionality

Given an application that fulfills the above assumptions, Sherpa maintains a central priority queue for each data item (or chunk of items). The queue is initialized so that each item has the same priority at the start of computation, and the items are ordered based on the semantic position in the dataset. Thus, if the priorities are never changed, the computation will proceed from whatever is defined as the "beginning" of the dataset to its "end." The computation executes one data item at a time, which has the highest priority but is not processed in the queue at the time of execution.

Starting from when the Sherpa application is launched, a concurrent computational engine will run computation based on the priority queue. A practical implementation will realize this engine either as a background, multi-threaded process, or on the server side.

The Sherpa interface includes a basic *steering panel*, modeled after work by Badam et al. [1], which provides simple steering controls that interface with the computational engine: starting, stopping, and pausing the computation. The panel also shows current progress.

3.3 Data Space View

Given a dataset with location semantics, the *data space view* is a spatial representation of the dataset. Depending on the application, the data view can be 1D or 2D in nature: for a gene sequence or timeline, for example, it would be represented by a single dimension, whereas for a geographic map or spatial data structure, it would be two-dimensional. A key aspect of the data view is that it communicates the position in the dataset using labels, ticks, or grid lines (or a combination of these), allowing users to orient themselves and navigate accurately in the spatial dimension. The view also contains a *summary visualization*, a *priority queue*, and a *progress indicator*.

3.4 Navigation Window

Finally, the *navigation window* on the data space view represents the user's attention on the dataset, which will guide the computational steering. It is represented by its *extents*: for a one-dimensional data space, this is a simple interval $[e_{min}, e_{max}]$, whereas for a two-dimensional one, it is a bounding box $[x_{min}, y_{min}, x_{max}, y_{max}]$. As such, the navigation window is initialized at the beginning of the exploration to cover the entire dataset $[0,1]$ or $[0,0,1,1]$ (inclusive).

Interacting with the navigation window can be done by *panning* (translating the extents) or *zooming* (changing the size of the window $e_{max} - e_{min}$). The main visualization window should be synchronized to always display only the portion of the data that is currently selected by the navigation window. Typically this is done by, for example, moving the window by dragging on the window itself using a mouse or finger touch (panning), changing window dimensions by dragging on one of the window borders using a mouse or finger (zooming), or changing the window size by rotating the mouse wheel or pinching (zooming). The extents will be kept in the range $[0,1]$.

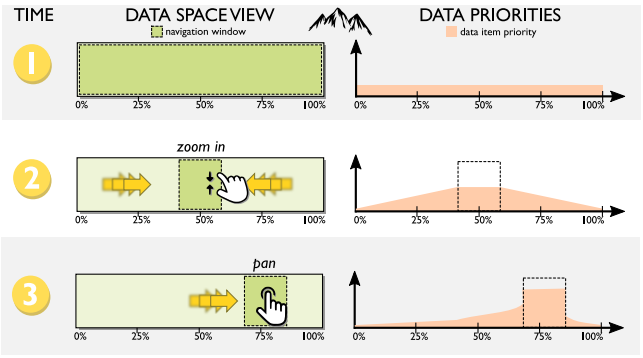

Figure 3: Mining attention as navigational behavior over time.

3.5 Mining Attention

The final piece is leveraging the user's attention. We use the navigational behavior of the user as a proxy for their attention. We base this on the intuition that the user's interaction with the navigation

window in data space is a representation of which part of the data the user is interested in. The behavior of the navigation window is then used to adjust the priority of each data item.

More specifically, we view attention as the position and dimension of the navigation window over time. Let us assume that the user confers a constant 1.0 units of attention on the view per time unit. If the entire dataset of N items (or segments of items) is within the navigation window, then each item will be receiving $1.0/N$ units of attention per time unit. No specific item will be receiving more attention than the others, leaving the priority queue unchanged. However, if the navigation window is zoomed, reducing its size to a smaller $n < N$, then all of the items still within the new navigation window will be receiving $1.0/n$ of attention per time unit. Numerically integrating this attention over time will enable Sherpa to essentially model user attention on the dataset (Figure 3).

Since attention may change over time, we also introduce a temporal decay function that gradually reduces the accumulated priority of each data item over time. We have experimented with several decay functions; the most useful is a radioactive decay function:

$$P(t) = P_0 \left(\frac{1}{2}\right)^{\frac{t}{t_{1/2}}}$$

where P_0 is the initial priority, t is the time parameter, and $t_{1/2}$ is the half-life of the priority decay. Values for specific constants will need to be determined for each application.

Finally, while we have not focused on collaborative aspects in this work, the method does allow for modeling the attention of multiple analysts based on their navigational behavior on the data space view. This will provide a mechanism for a team to collectively steer the computation. However, the accumulation of attention over time may have to be modified to prevent one user from gaming the system by shrinking their navigation window to incur a high attention on a small part of the dataset, thus prioritizing only their view.

4 SHERPA FOR GENOMICS DATA

To showcase the Sherpa framework, we developed an interactive visual analytics tool—SHERPA GENE—that uses Sherpa to support attention-based computational steering in functional genomics (Figure 1). This tool fulfills the general Sherpa requirements as follows:

- *Dataset:* We use genomics data, gene expression, and DNA methylation indexed by location within the human genome.
- *Computation:* Our user group is interested in understanding mechanisms in which DNA methylation regulates the expression of genes of interest in cancer and corresponding normal tissue, and whether these mechanisms are consistent across different tumor types. To understand these mechanisms, statistical inferences are used based on measuring correlation between DNA methylation and gene expression within specific tissues (understanding mechanism within tissue), correlation between expression or DNA methylation across tissues (understanding the consistency of mechanism across tissues), and overlap of regions of differential methylation in cancer (understanding the consistency of mechanism across tissues). The computations required to calculate these statistical measures of biological importance are easily parallelizable.
- *Visualization:* We use several progressive visualizations that summarize the current focused region: a genes track indicating specific genes contained in the region of interest, a gene expression heatmap showing similarity (and dissimilarity) of expression for multipe tissues, scatterplots showing trends in expression or DNA methylation within and across multiple tissues, line tracks showing DNA methylation values at their corresponding genomic position, and region tracks showing regions of differential methylation in different tumor types from which the overlap of these regions of interest can be observed.

The interactive workflow of Sherpa Gene typically involves exploring specific regions of interest. Therefore, we use the user's genomic location within the chromosome to steer the computation.

4.1 Dataset

Sherpa Gene contains human transcriptome data from the Gene Expression Barcode Project [26] for 105 different tumor and normal tissues. The database also contains methylation signal data [36] for 6 different tissue types and includes both cancer and tumor samples. We selected four tissue types in the implementation: colon, thyroid, breast, and lung, across two different conditions: tumor and normal. Overall, the database contains over 50,000 rows of gene expression data per gene and over 480,000 rows of DNA methylation data at specific locations in the human genome. We also include regions of differential DNA methylation between tumor and corresponding normal tissue (referred to as "blocks"). The number of blocks range from 1,000 to 2,000 regions across different cancer types.

4.2 Computational Algorithms

Our data includes three data modalities indexed by genomic location: gene expression, DNA methylation at specific locations, and blocks of differential DNA methylation between tumor and corresponding normal tissue. While there are many types of computations that can be applied to such data, Sherpa Gene implements the following:

- **Promoter DNA Methylation-Gene Expression Correlation:** Correlation between the DNA methylation and gene expression of specific tissues (normal and tumor). DNA methylation is the best understood epigenetic mechanism of gene regulation. Measuring the correlation between DNA methylation and expression in a specific tissue provides insight into this mechanism for specific genes in a tissue of interest.
- **Methylation Block Overlaps:** Identifying genomic regions where DNA methylation is statistically different between tumor and corresponding normal tissue is essential to understand the role of DNA methylation in cancer. Once these regions of interest are identified for each tumor type of interest, computing the overlap of these regions across tissues provides insight about the consistency, or uniqueness, of this mechanism across cancer types, which is a characteristic of biological importance. We compute the ratio of block overlap between pairs of tissue for a specific genomic region.
- **Gene Expression or DNA Methylation Correlation:** Correlation between gene expressions or DNA methylation between pairs tissues within a genomic region. This indicates similarity in gene regulation between tissues. Similarities of interest are those tumor types that show similar gene regulation, as well as normal tissues that show similar gene regulation.
- **t-test for Differential Expression or Differential Methylation:** We also compute a t-statistic for expression or DNA methylation between pairs of tissue within a genomic region. We do this to measure the dissimilarity in gene regulation across pairs of tissues. As above, dissimilarities of interest are those tumor types that show different gene regulation, as well as normal tissues that exhibit different gene regulation.

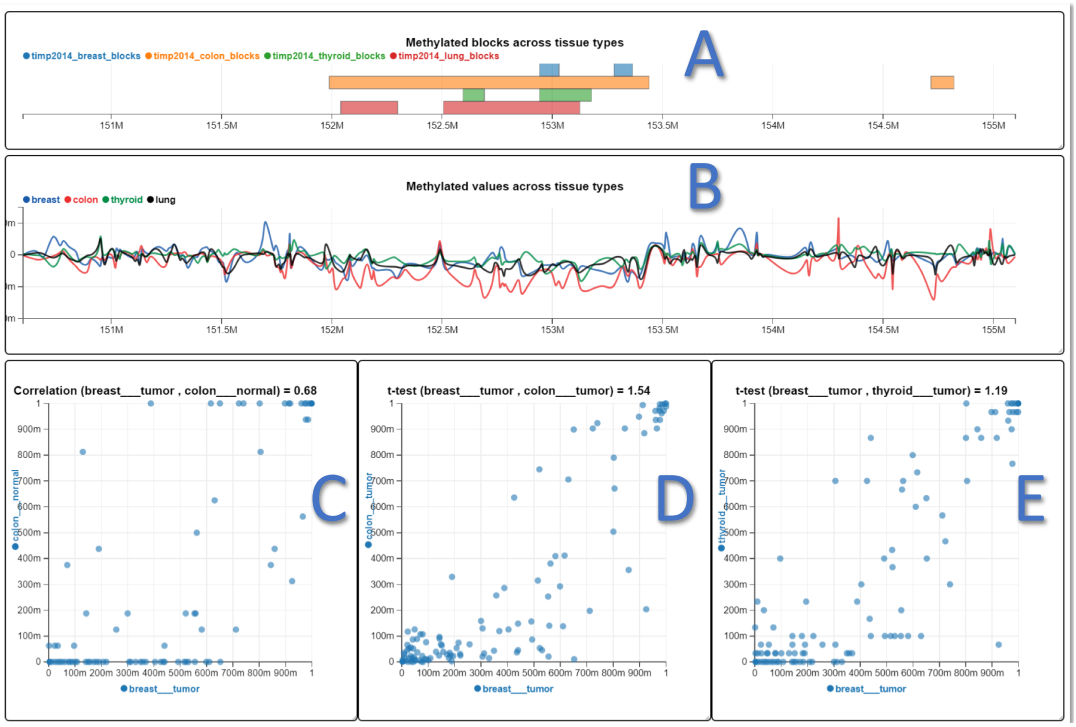

Figure 4: Example charts from Sherpa Gene (also see Figure 1 for the genes track and heatmap for gene expression across breast, colon, thyroid, and lung tissues). (A): Methylated Block track: indicating differentially methylated genomic regions within which DNA methylation is significantly different (according to an offline statistical inference) between tumor and the corresponding normal tissue for the four tissues under study; (B): Methylation line track: shows the difference in DNA methylation between tumor and corresponding normal tissue at specific genomic positions; (C): Scatterplot of gene expression for two different tissues, illustrating correlation between gene activity in those tissues; (D, E): Scatterplot of gene expression for two tissues, illustrating difference of gene expression in those two tissues measured by a t-statistic.

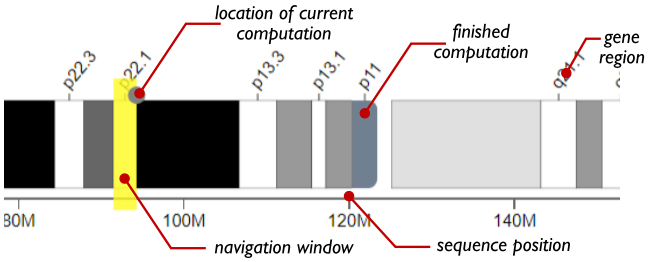

Figure 5: Detail of a chromosome ideogram—an idealized depiction of a chromosome—used as a data space view in Sherpa Gene. Navigating the chromosome will steer the server-side computation.

4.3 Steering Interface

Our prototype implements the Sherpa steering interface at the bottom of the display. The steering control panel (Fig. 1) allows for starting and stopping the server-side computation. The data space view is implemented as a chromosome ideogram (Figure 5), which is an idealized graphic representation of a chromosome. The navigation window is a yellow region showing the current focus. A progress bar shows the current status, which will gradually begin to fill in with a transparent blue color as the computation proceeds.

Moving the navigation window on the data space ideogram will both steer the computation as well as govern which visualizations will be shown in the main view (see below). Pilot testing caused us to immediately give regions inside the navigation window in Sherpa Gene top priority. If the user does not navigate, or if the computation for a specific focus region has finished, the computation will continue on other genomic regions based on accumulated priorities.

4.4 Progressive Visualization

The main view of Sherpa Gene is consumed by progressive visualizations that show the currently selected genomic region of focus. Additional charts showing results are added as they are produced.

More specifically, the genes track (Fig. 1) is shown in the top of the main view to provide an overview of the genes within a region. A heatmap (Fig. 1) with cell and tissue types as rows, and genes as columns allows for comparing expression values. Methylation block overlaps (Fig. 4A) are shown in a stacked blocks track for all tumor types. A DNA methylation values line track (Fig. 4B) makes it possible to investigate changes and trends in detail. The space below these charts is used for adding scatterplots (Fig. 4C, D, E), one by one, each representing correlations in expression or DNA methylation between normal and tumor tissue types.

4.5 Implementation Notes

The Sherpa Gene implementation uses a client/server architecture. The client interface was developed with modern web technology: HTML5, JavaScript, and CSS3, along with Polymer 2.0 [31] and the Epiviz web component library [21]. The backend server consists of (1) a MySQL database, which stores the genomic data; (2) the Epiviz data provider [10], which extracts data from database; and (3) a computational server, that runs all the computations and provides a websocket endpoint using the Python Flask framework. The data provider ensures fast retrieval of the data from the MySQL server, and the websocket connection enables streaming results back from server to client through chunks when one batch is finished.

5 Qualitative Evaluation

The goal of the Sherpa toolkit is to enable an analyst to steer a computational process using their navigation alone. Thus, we are building on the notion of progressive visual analytics [11], which does include both visual updates (output) as well as computational steering (input), but which does not stipulate *how* computation should be guided. Our hypothesis is that the interest-based computational steering (progressive input and output) that Sherpa embodies will perform better than just gradually updating the visualization (progressive output only). To test this hypothesis, we conducted a qualitative expert review [37] using our Sherpa Gene implementation.

5.1 Participants

We recruited in total 5 participants (4 male, 1 female): 2 from a visualization group and 3 from a bioinformatics lab at our university. Participants were between 24 and 33 years of age, had normal or corrected-to-normal vision, and were experienced computer users. In particular, all participants had significant experience in visualization, bioinformatics, or computer science, had research training, and were well-versed in visualization and genomics.

5.2 Experimental Design

We organized each session into three within-participant conditions:

- *Blocked:* In this condition, the computation was completed prior to a trial commenced, thus giving the user immediate access to the full results. The participants were merely informed of the full execution time required to perform this computation (5-6 minutes depending on genome size); they were not required to wait for the duration of this time. The experimental software used was our genomics prototype application—Sherpa Gene, as described in the previous section—but with progressive and steering functionality disabled.

- *Progressive output:* Here, the computation was launched at the same time as the trial was started, but the Sherpa attention-based steering functionality in our tool was disabled. Thus, the computation proceeded from the beginning of the genome until it reached its end (which, as stated above, took approximately 5-6 minutes). During this time, the main visualizations in the genomic application were progressively updated, and participants could interact with the tool to perform tasks. Participants could use the data space view and navigation window to move around the dataset, but their navigation was not used to modify priorities; i.e., there was no computational steering support.

- *Progressive output & input:* In this condition, we enabled the full Sherpa functionality, including attention-based computational steering. Computation started simultaneously with the trial, and participants had full access to all features of the tool.

5.3 Task and Procedure

For the purposes of the expert review, we asked our participants to answer a collection of five tasks related to a specific chromosome. With three conditions, we created three separate such sets of tasks for three different chromosomes. These were balanced between the three conditions, but the order of conditions was always the same to enable precomputation to finish prior to each session for the initial *blocked* condition. Given that our study is qualitative, we think that the lack of counterbalancing had little impact on our results. In fact, presenting the non-progressive version first, where all data is immediately available, provides a useful baseline comparison.

Participants were encouraged to solve tasks in any order. We asked participants to follow a think-aloud protocol, and recorded their utterances. The experimenter took extensive notes, and the software stored an interaction log. Each session lasted 45–55 minutes.

The tasks were exclusively location-based in nature, i.e., about genomic regions containing a specific gene of interest that a participant could navigate to using the data view. While this certainly favors the Sherpa method, where navigational behavior will affect computation order, this was precisely the purpose of our expert review. We wanted to understand the utility of this method rather than study completion time and task accuracy for a fully ecologically valid use case. We leave such studies for future work.

6 Results

We first report results from the evaluation as well as the think-aloud comments, then describe observations and post-study interviews.

6.1 Performance Results

All five participants successfully finished the tasks in all three experimental conditions. When first starting the application, participants all experimented with the data space view and navigation window to understand the steering functionality. They were pleased to see results gradually update as computation proceeded in the background. One participant said, *"I don't care about how the computation works, but I think showing intermediate results is absolutely necessary."*

Compared with the progressive output condition, the Sherpa functionality gave participants more perceived control over the visualizations, thus making it easier to access the results. While we did not measure the time for individual tasks, we observed that participants spent less time in total to finish the tasks in the Sherpa condition than with progressive output. Three participants said they were frustrated when they realized there was no interactive steering in this condition. With respect to the blocked condition, where computations were pre-completed, the overall usage time for finishing all five tasks was only slightly faster than the Sherpa condition.

6.2 Usability Feedback

Overall, participants were all very interested in the Sherpa framework and praised our effort at combining computational steering and visual analytics. All participants thought Sherpa Gene was very useful for exploratory analysis, and 2 participants said that it was even more helpful for search tasks, in which one needs to explore multiple regions within the data, such as *"find the region that has the highest correlation between colon tumor and normal tissue."* One participant mentioned that the prototype application *"makes me motivated to control the computation,"* essentially forming an analytical partnership between the user and the computer [6]. When given a task where participants needed to look into multiple regions to find the answer, e.g., a search task, they would navigate to those regions and get familiar with the results, which would be progressively computed based on the navigation. In other words, the use of attention as a prompt conformed well with our participant's intuition when foraging for information [30]. Furthermore, one participant said, *"different orders of exploration [computations] may produce more insights, which users can control easily [in Sherpa]."* From our observations, we also saw that participants often selected diverse regions seemingly at random (many not related to the task) to merely gain understanding about the data.

As for the conditions in the study, all five participants thought the Sherpa condition was the superior one. Three stated that the blocked condition with pre-loaded computations was not appropriate because a common task is to just get a quick view of a small region in the dataset, and they would not want to wait for all computation to finish. One noted that pre-loading all computations in one shot is not feasible. Pre-computing results may also incur unnecessary waiting time since different tasks may need different computations.

Surprisingly, the progressive output condition was the least preferred condition. One participant claimed that he would not want to use a tool without steering: *"When I select a region, I'd like to see the results [in that region], that's the purpose of my selection."*

One participant also suggested the tool would be useful in the understanding of how disease correlates with gene regulations, where analysts would navigate to genes with similar functionality on the same disease but which may not be located closely w.r.t. position.

6.3 Points of Improvements

Participants also provided valuable suggestions on how our tool can be improved. Two participants thought that it would be useful to maintain a history of user-selected regions. This may be particularly helpful for complex tasks that require comparing data across multiple regions. In a way, our numerical integration of priority over time does serve this purpose, as it will "remember" parts of the data space the user has visited, and prioritize their computation.

Beyond that, some participants felt that our visualizations easily become difficult to understand when the corresponding computations are complex. While not strictly related to our method, it is true that progressively updating visualizations exacerbates such complexity.

Finally, one participant also raised a concern about the trade-offs between how much computational power the user wants to leverage and how fast steering should work. While this is an interesting question, it is beyond the scope of this paper.

7 Sherpa for Stock Market Data

We also present SHERPA STOCK, which applies Sherpa to attention-based market indicators for massive stock market data (Figure 6):

- *Dataset:* We use market data for a set of stocks on a daily basis, including the stock name, price at opening/closing, high and low prices during the day, and volume traded. The temporal dimension is the Sherpa location. Our example here uses an S&P 500 dataset,[1] but obviously any such data can be used.
- *Computation:* We calculate several standard market indicators on a daily basis using the dataset: moving averages, autocorrelation with specific lags, and cross-correlations between the pairs of stocks. In order to fulfill the parallelizability requirement, our calculations must be independent of calculations for other time points in the dataset. While this is not a concern, it also means that several optimizations (for example, a global moving average) cannot be used. With 500 stocks over 5 years, our calculations usually take 8-10 seconds per day to complete.
- *Visualization:* Our implementation uses a single progressive line graph visualization for different market indicators as well as stock prices (Figure 6). The line graph will progressively fill in with visual data points as the computations are completed.

Stock market analysis is often closely tied to positions in time. For this reason, we use temporal navigation to steer the computation.

7.1 Steering Interface

The Sherpa steering interface in Sherpa Stock is located at the bottom of the display (Figure 7). The data space view essentially provides an overview of the full time period covered by the dataset as a timeline. The navigation window is a light gray rectangle on top of the data space, and moving or resizing this rectangle—by dragging or rolling the mouse wheel—will change the position or extents of the data in the main visualization window. In other words, using the "overview+detail" model [15], the data space is the *overview*, and the main line graph is the *detail*.

The data space shows a low-fidelity version of the stock market index over the full time period. Furthermore, the data space will show the actual plot for regions that the computations have finished, an approximation line for those that the computations have not. The exact order of the computation is obviously controlled by the Sherpa attention-based steering mechanism, but the entire data space will eventually be filled in with exact data points when the computation has finished for the whole dataset.

[1] https://www.kaggle.com/camnugent/sandp500

7.2 Implementation Notes

Sherpa Stock uses the same basic Sherpa framework and architecture as Sherpa Gene, including a client/server architecture based on HTML5, JavaScript, and CSS3. The frontend uses Google Polymer 2.0 [31] for user interface and web components. The backend server communicates with the client using a Python Flask interface. All computations on stock market indicators are performed using a computational engine built in Python. The prototype uses a stock market dataset stored as a local (uploaded) file; a realistic implementation would instead query a financial data provider for full flexibility.

8 Discussion

Here we attempt to explain our results for implicit computational steering and then discuss limitations of our work.

8.1 Explaining the Results

In the user study with Sherpa Gene, the framework provided a significant advantage for participants solving location-based tasks, particularly when the task involves searching through multiple regions. Compared with only progressive visualization and pre-computed conditions, participants were more engaged in the exploratory data analysis process in the Sherpa condition. This is not surprising: steering, even implicitly using navigation behavior, provides direct control over the computation. With no steering, participants could not see the outputs for a region until computation was finished.

However, we were somewhat surprised to see only a small difference between Sherpa and the pre-computed condition—the gold standard, where all results were immediately available. One explanation is that in Sherpa Gene, the individual computations are lightweight and can be finished quickly, which means that navigating to a specific region will quickly yield results. Initial results would come in within just half a minute, which would not be much slower than for the precomputed condition. A more time-consuming server-side computation would not yield the same responsiveness.

Another surprising observation is that all participants preferred the blocked over the progressive output condition. This indicates that interaction is an essential part of PVA. These results cause us to speculate that progressive visualization without steering may actually have a negative effect on user experience. Of course, we did not compare the output-only condition to a truly blocked condition where the participants had to sit through a progress bar slowly filling up for the entire duration of the computation.

8.2 Limitations

As mentioned before, the tasks in our evaluation were all location-based questions. This was intentional to elicit findings specifically about Sherpa's steering functionality, but means that the study is not fully representative of realistic genomics workflows. Future studies should include more general and ecologically valid tasks.

While we are using a real-world genomics dataset [26], we selected our computations to be parallelizable so that they would fit within the Sherpa framework, which is certainly not true of all algorithms used for functional genomics. Nevertheless, we believe they were complex enough to generate realistic exploratory tasks that enabled studying the utility of Sherpa. Besides, the visualization in Sherpa Gene draw from the Epiviz framework [21], which has been proven to be easily scalable and reusable to other genomics datasets.

As a result, however, our evaluation using Sherpa Gene involved computations that merely lasted 5-6 minutes in total to finish. This was a deliberate choice to balance the various conditions: it would have been quite frustrating for our participants if we had forced them to wait up to, say, 30 minutes to complete the trials in the *progressive*

Figure 6: Example of the Sherpa Stock application being used to calculate market indicators for 10 years of the S&P 500 stocks. This image shows three visualizations within the focused region: S&P 500 index, Apple, and Google stock price. The bottom overview visualization is the data space view that user controls the gray navigation window.

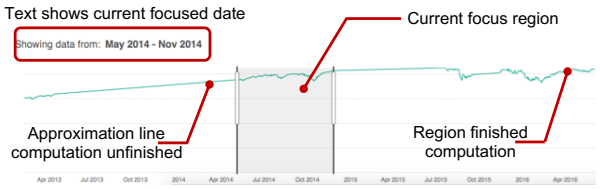

Figure 7: Detail of the Sherpa Stock timeline. The line graph visualization shows a stock market index—the S&P 500 in this case—as an abstraction of the stock market behavior over time. The gray rectangle shows the position of the navigation window. Unfinished regions will show a straight line as an approximation.

output condition. Needless to say, it is trivial to add on a significant amount of computation for genomic data such as this. In fact, our Sherpa Gene tool is built to be modular and extensible so that it can easily be customized with the computations our users require.

You could even argue that a more realistic setting would require participants in the *blocked* condition to sit and stare at a progress bar while the computation completed. Such an approach is, after all, currently dominant in tools such as this. Again, we chose to skip the waiting stage entirely and instead merely presented the final result in the interest of making our participants' experience less frustrating.

Finally, utilizing navigation behavior for computational steering is susceptible to a variant of the "Midas touch" problem in HCI [40]: distinguishing between interaction for exploration (implicit) vs. interaction for selection (explicit). Put differently, some navigation behavior may not be a direct indication of interest, but rather merely a form of *epistemic* action [22] (cf. *pragmatic* ones) that helps the analyst get an overview of the dataset. We saw indications of this in our study: some participants would idly "click around" on the ideogram bar to view various regions. We see this as a caution against attempting to infer too much from navigation behavior alone.

9 Conclusion and Future Work

While the emergent research topic of progressive visual analytics (PVA) provides an exciting, realistic, and future-proof method for managing even massive datasets in an interactive workflow [11, 47], existing PVA systems have—with a few exceptions [1, 14]—largely left the input side of the equation unexplored. To remedy this, we have proposed the Sherpa method for computational steering in PVA based on user navigation, thus eliminating the need for the analyst to explicitly control the computation order. We implemented two use case examples with genomic data (Sherpa Gene) and stock market data (Sherpa Stock). Results from our expert review with bioinformaticians using Sherpa Gene for genomics analysis provide empirical validation for our approach; while obviously having immediate access to computational results is preferable, our participants felt that the Sherpa model was more empowering and efficient than merely seeing progressive visual updates. We also presented another illustrative example of how the Sherpa model could be applied to financial stock market data evolving over time.

In the future, we intend to explore the Sherpa model further, including applying it to new datasets and computations, integrating it with an insight notification system [6], and investigating additional implicit computational steering mechanisms beyond navigation. We are also interested in studying how mining attention using Sherpa can be best realized for collaborative data analysis.

Acknowledgments

This work was partially supported by the U.S. National Institutes of Health grant R01GM114267. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] S. K. Badam, N. Elmqvist, and J.-D. Fekete. Steering the craft: UI elements and visualizations for supporting progressive visual analytics.

Computer Graphics Forum, 36(3):491–502, 2017. doi: 10.1111/cgf.13205

[2] M. Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 446–453. ACM, New York, NY, USA, 2000. doi: 10.1145/332040.332473

[3] M. Beaudouin-Lafon. Designing interaction, not interfaces. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pp. 15–22. ACM, New York, NY, USA, 2004. doi: 10.1145/989863

[4] J. Biddiscombe, J. Soumagne, G. Oger, D. Guibert, and J.-G. Piccinali. Parallel computational steering and analysis for HPC applications using a ParaView interface and the HDF5 DSM virtual file driver. In *Proceedings of the Eurographics Conference on Parallel Graphics and Visualization*, pp. 91–100. Eurographics Association, Geneva, Switzerland, 2011. doi: 10.2312/EGPGV/EGPGV11/091-100

[5] E. Bier, M. Stone, and K. Pier. Enhanced illustration using magic lens filters. *IEEE Computer Graphics and Applications*, 17(6):62–70, Nov./Dec. 1997. doi: 10.1109/38.626971

[6] Z. Cui, S. K. Badam, A. Yalçin, and N. Elmqvist. DataSite: Proactive visual data exploration with computation of insight-based recommendations. *Information Visualization*, 18(2):251–267, 2019. doi: 10.1177/1473871618806555

[7] H. Doleisch, H. Hauser, M. Gasser, and R. Kosara. Interactive focus+context analysis of large, time-dependent flow simulation data. *Simulation*, 82(12):851–865, 2006. doi: 10.1177/0037549707078278

[8] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 473–482. ACM, New York, NY, USA, 2012. doi: 10.1145/2207676.2207741

[9] A. Endert, S. Fox, D. Maiti, and C. North. The semantics of clustering: analysis of user-generated spatializations of text documents. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pp. 555–562. ACM, New York, NY, USA, 2012. doi: 10.1145/2254556.2254660

[10] Epiviz data provider, 2018. doi: 10.5281/zenodo.1422712

[11] J.-D. Fekete. ProgressiVis: a toolkit for steerable progressive analytics and visualization. In *Proceedings of the IEEE VIS Workshop on Data Systems for Interactive Analysis*, p. 5. IEEE, Piscataway, NJ, USA, 2015.

[12] D. Fisher, R. DeLine, M. Czerwinski, and S. M. Drucker. Interactions with big data analytics. *ACM Interactions*, 19(3):50–59, 2012. doi: 10.1145/2168931.2168943

[13] D. Fisher, I. Popov, S. Drucker, and m. c. schraefel. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 1673–1682. ACM, New York, NY, USA, 2012. doi: 10.1145/2207676.2208294

[14] M. Grusky, J. Jahani, J. Schwartz, D. Valente, Y. Artzi, and M. Naaman. Modeling sub-document attention using viewport time. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 6475–6480. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025916

[15] K. Hornbaek, Bederson, B. B., and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction*, 9(4):362–389, 2002. doi: 10.1145/586081.586086

[16] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 159–166. ACM, New York, NY, USA, 1999. doi: 10.1145/302979.303030

[17] E. J. Horvitz. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine*, 28(2):3, 2007. doi: 10.1609/aimag.v28i2.2036

[18] D. J. Jablonowski, J. D. Bruner, B. Bliss, and R. B. Haber. VASE: The visualization and application steering environment. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pp. 560–569. IEEE, Piscataway, NJ, USA, 1993. doi: 10.1109/SUPERC.1993.1263505

[19] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999. doi: 10.1145/331499.331504

[20] J. Jo, J. Seo, and J.-D. Fekete. PANENE: A progressive algorithm for indexing and querying approximate k-nearest neighbor. *IEEE Transactions on Visualization and Computer Graphics*, PP(1):1–14, 2018. To appear. doi: 10.1109/TVCG.2018.2869149

[21] J. Kancherla, A. Zhang, B. Gottfried, and H. C. Bravo. Epiviz web components: reusable and extensible component library to visualize functional genomic datasets. *F1000Research*, 7, 2018. doi: 10.12688/f1000research.15433.1

[22] D. Kirsh and P. P. Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18(4):513–549, 1994. doi: 10.1207/s15516709cog1804_1

[23] S. Ko, J. Zhao, J. Xia, S. Afzal, X. Wang, G. Abram, N. Elmqvist, L. Kne, D. V. Riper, K. P. Gaither, S. Kennedy, W. J. Tolone, W. Ribarsky, and D. S. Ebert. VASA: Interactive computational steering of large asynchronous simulation pipelines for societal infrastructure. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1853–1862, 2014. doi: 10.1109/TVCG.2014.2346911

[24] D. Lagun and M. Lalmas. Understanding user attention and engagement in online news reading. In *Proceedings of the ACM Conference on Web Search and Data Mining*, pp. 113–122. ACM, New York, NY, USA, 2016. doi: 10.1145/2835776.2835833

[25] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2122–2131, 2014. doi: 10.1109/TVCG.2014.2346452

[26] M. N. McCall, H. A. Jaffee, S. J. Zelisko, N. Sinha, G. Hooiveld, R. A. Irizarry, and M. J. Zilliox. The gene expression barcode 3.0: Improved data processing and mining tools. *Nucleic Acids Research*, 42(D1):D938–D943, 2013. doi: 10.1093/nar/gkt1204

[27] J. D. Mulder, J. J. van Wijk, and R. van Liere. A survey of computational steering environments. *Future Generation Computer Systems*, 15(1):119–129, Feb. 1999. doi: 10.1016/S0167-739X(98)00047-8

[28] S. G. Parker and C. R. Johnson. SCIRun: A scientific programming environment for computational steering. In *Proceedings of the ACM/IEEE Conference on Supercomputing*. IEEE, Piscataway, NJ, USA, 1995. doi: 10.1145/224170

[29] N. Pezzotti, B. P. F. Lelieveldt, L. van der Maaten, T. Hollt, E. Eisemann, and A. Vilanova. Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, July 2017. doi: 10.1109/TVCG.2016.2570755

[30] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of the International Conference on Intelligence Analysis*, vol. 5, pp. 2–4. The MITRE Corporation, McLean, VA, USA, 2005.

[31] Polymer. `https://www.polymer-project.org/`, 2019.

[32] H. Ribicic, J. Waser, R. Fuchs, G. Bloschl, and E. Gröller. Visual analysis and steering of flooding simulations. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):1062–1075, 2013. doi: 10.1109/TVCG.2012.175

[33] K. Singh and R. Balakrishnan. Visualizing 3D scenes using non-linear projections and data mining of previous camera movements. In *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pp. 41–48. ACM, New York, NY, USA, 2004. doi: 10.1145/1029949.1029956

[34] A. Skopik and C. Gutwin. Improving revisitation in fisheye views with visit wear. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 771–780. ACM, New York, NY, USA, 2005. doi: 10.1145/1054972.1055079

[35] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society Press, Los Alamitos, CA, USA, 2005.

[36] W. Timp, H. C. Bravo, O. G. McDonald, M. Goggins, C. Umbricht, M. Zeiger, A. P. Feinberg, and R. A. Irizarry. Large hypomethylated blocks as a universal defining epigenetic alteration in human solid tumors. *Genome Medicine*, 6(8):61, 2014. doi: 10.1186/s13073-014-0061-y

[37] M. Tory and T. Möller. Evaluating visualizations: Do expert reviews work? *IEEE Computer Graphics and Applications*, 25(5):8–11, 2005. doi: 10.1109/MCG.2005.102

[38] J. W. Tukey. *Exploratory Data Analysis*. Pearson, Reading, MA, USA, 1977.

[39] R. van Liere, J. D. Mulder, and J. J. van Wijk. Computational steering. *Future Generation Computer Systems*, 12(5):441–450, Apr. 1997. doi: 10.1016/S0167-739X(96)00029-5

[40] B. M. Velichkovsky, A. Sprenger, and P. Unema. Towards gaze-mediated interaction: Collecting solutions of the "Midas touch problem". In *Proceedings of the INTERACT Conference*, pp. 509–516. Springer, Boston, MA, USA, 1997. doi: 10.1007/978-0-387-35175-9_7

[41] J. Vetter and K. Schwan. Progress: A toolkit for interactive program steering. In *Proceedings of the International Conference on Parallel Processing*, pp. 139–142. CRC Press, Boca Raton, USA, 1995.

[42] J. Vetter and K. Schwan. High performance computational steering of physical simulations. In *Proceedings of the International Parallel Processing Symposium*, pp. 128–132. IEEE, Piscataway, NJ, USA, 1997. doi: 10.1109/IPPS.1997.580866

[43] J. S. Vetter. Computational steering annotated bibliography. *SIGPLAN Notices*, 32(6):40–44, 1997. doi: 10.1145/261353.261359

[44] J. Waser, R. Fuchs, H. Ribicic, B. Schindler, G. Bloschl, and E. Gröller. World lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, 2010. doi: 10.1109/TVCG.2010.223

[45] J. Waser, H. Ribicic, R. Fuchs, C. Hirsch, B. Schindler, G. Bloschl, and E. Gröller. Nodes on ropes: A comprehensive data and control flow for steering ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1872–1881, 2011. doi: 10.1109/TVCG.2011.225

[46] Y. L. Wong, J. Zhao, and N. Elmqvist. Evaluating social navigation visualization in online geographic maps. *International Journal of Human-Computer Studies*, 31(2):118–127, 2015. doi: 10.1080/10447318.2014.959106

[47] E. Zgraggen, A. Galakatos, A. Crotty, J.-D. Fekete, and T. Kraska. How progressive visualizations affect exploratory analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1977–1987, 2017. doi: 10.1109/TVCG.2016.2607714

[48] J. Zhao, D. Wigdor, and R. Balakrishnan. TrailMap: facilitating information seeking in a multi-scale digital map via implicit bookmarking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 3009–3018. ACM, New York, NY, USA, 2013. doi: 10.1145/2470654.2481417