ModelSpace: Visualizing the Trails of Data Models in Visual Analytics Systems

Category: Research

Paper Type: algorithm/technique

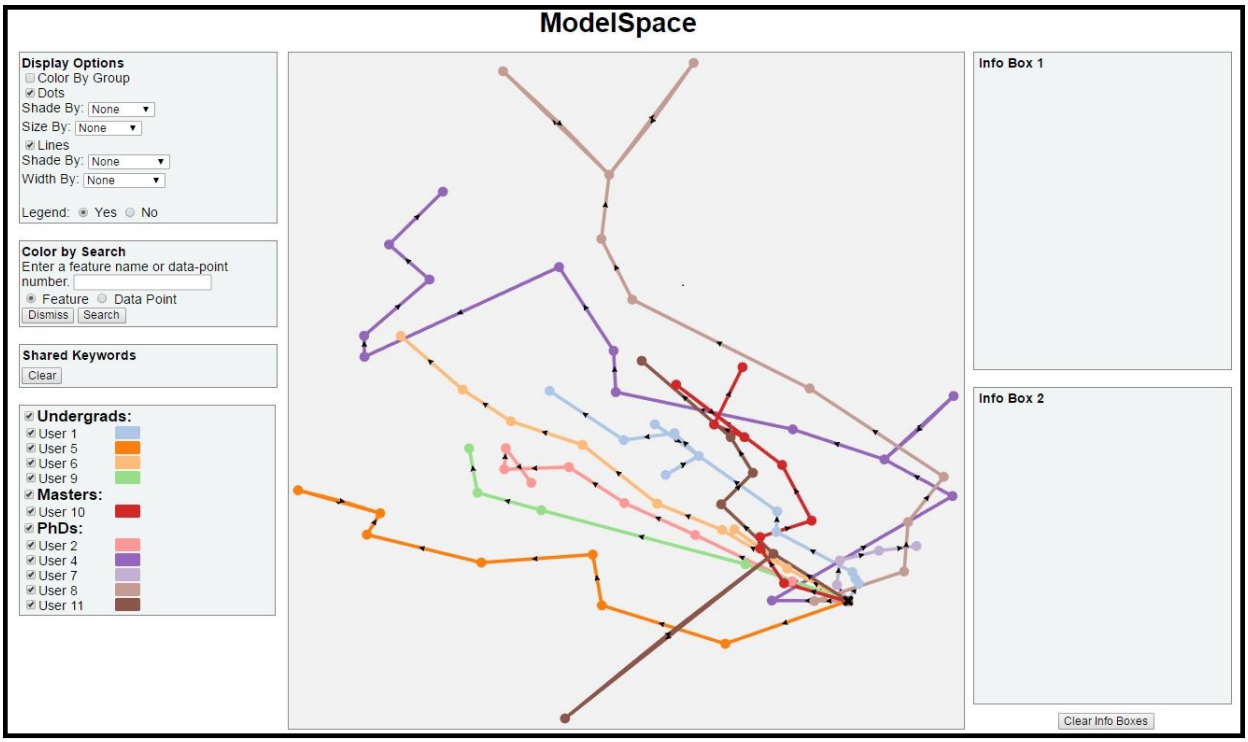


Fig. 1: The ModelSpace prototype tool

Abstract—User interactions with visualization systems have been shown to encode a great deal of information about the users' thinking process, and analyzing their interaction trails can teach us more about the users, their approach, and how they arrived at insights. This deeper understanding is critical to improving their experience and outcomes, and there are tools available to visualize logs of interactions. It can be difficult to determine the structurally interesting parts of interaction data, though, like what set of button clicks constitutes an action that matters. In the case of visual analytics systems that use machine learning models, there is a convenient marker of when the user has significantly altered the state of the system via interaction: when the model is updated based on new information. We present a method for *numerical analytic provenance* using a series of numerical representations of the changing state of a system. Leveraging this numerical representation, we apply high-dimensional visualization to show and compare user trails through the space of possible states (i.e. models). We evaluate this approach with a prototype tool, ModelSpace, applied to two case studies on experimental data from model-steering visual analytics tools. ModelSpace reveals the users' individual progress, the relationships between their paths, and the characteristics of certain regions of the space of possible models.

Index Terms—Numerical analytic provenance; visual analytics; analytic provenance; interaction history; user interaction; visualization; machine learning; evaluation

1 Introduction

Visual analytics facilitates discovery and analytical reasoning via the combination of data analytic models and interactive visualizations [41]. Because such systems provide tight connections between the user, the visual interface and the underlying analytics, the user's interactions within visual analytics systems have been found to contain a great deal of information about the users' thinking processes, their approaches, and how they arrive at insights [16]. The design of automated and semi-automated methods for recovering such information by analyzing the user's interaction history and analysis trails – commonly referred to *analytic provenance* – has become an increasingly important research topic in the visualization community due to its importance in training and verification, and to its role in the development of mixed-initiative systems [36, 37].

However, many existing tools in analytic provenance only go so far as to show a record of the user's interactions (e.g., [38, 25, 22, 28]). They seldom contain methods of visualizing the intermediate software states or data models typically generated behind-the-scenes by visual analytics systems. For visual analytics, this often entails the sequence of different parameters assigned to the analytic models to show specific aspects of the data to foster exploration and analysis. In addition, they rarely communicate the logical link between the interactions and the resulting models. Many methods of understanding a user's analytic

provenance typically involve a tedious manual reading of the logs as in Dou et al. [16], or building a system that codes its own interactions into a taxonomy so manual review is more convenient [21, 25]. Generally, the effort required to synthesize and analyze these logs is a bottleneck to studying analytic provenance. More recently there are visual analytics sytems that help to discover patterns within logs by enabling grouping or searching of user actions [23, 1, 12].

In this paper we propose an alternate, visual approach to analytic provenance that is designed for the growing number of systems using machine learning (though may be flexible enough to be used on many other types), but automatic enough not to require manually reading full logs. We create a mathematical representation of users' progress in using software, introducing *numerical analytic provenance*. By creating a vector space to represent software states (possibly extracted from logs), we can visualize users' process of using an analytic tool. Each software state corresponds to some view of the data delivered to the user in response to some input or controls. Conveniently, systems that leverage machine learning build models as users interact and those can easily be converted to vectors. Therefore, visualizing sequences of these models means seeing the progress of users through their analytic process. By gathering these and creating a vector representation, we can visualize the user's progress through the space of possible states, i.e. the provenance of their analysis.

We have implemented the numerical analytic provenance approach in a prototype tool called *ModelSpace*. This tool visualizes the analytic trail of a user by creating a spatial layout of the visual system states, using their vector form. A given user's trail is connected with a line and color-coded, providing a *connected scatterplot* [24], where the points represent states and the lines connecting the points represent the transitions between the states. The layout is also similar to Time Curves [2] because we have a sequence of vectors connected to show progress through time.

With such a compact representation, ModelSpace can visualize multiple users' analysis trails, or multiple analysis trails of the same user in the same canvas. In this way, we visualize how users incrementally interact with and change the analytic models in visual analytics systems. Further, the model spatialization provides a baseline structure that we annotate with data about interactions between state changes. In this view, analysis trails can be quickly compared and analyzed. For example, when two trails include adjacent states, it may signify that these two investigations came to similar inquiries, reflected by the similar models the users were considering. ModelSpace provides features to deepen the exploration, like the ability to highlight visual elements with a keyword search over the interaction details.

We tested our prototype on data collected from experiments with two visual analytics systems: (1) Dis-Function [8], an interactive tool for learning models about high-dimensional numerical data by simply performing iterative tweaks to a data visualization, and (2) Doc-Function [7], a tool that allows sense-making of text corpora through manipulation of keyword spatializations based on their perception of keyword relationships. The authors of those works provided the interaction logs and other data collected during the evaluation experiments of those software prototypes. Our ModelSpace implementation parses the logs (with a custom function per application), extracts the states, and provides an interactive visualization that makes it possible to explore a wide collection of facets of the participants' analytic provenance and develop insights into how different users explored the data.

While, we performed our experiments on two visual analytics systems with machine learning back-ends that lent their internal state well to numerical analytic provenance, we posit that the use of ModelSpace can be extended to other, non-visual-analytics platforms. Recommender systems, for example, are not analytics systems, but export models at each step of user interaction that could be visualized and compared with numerical analytic provenance. In the Discussion, we explore an application to a system that uses a visual interface but has no back-end machine learning model. We also discuss the limitations of our approach and current prototype, and describe areas for future improvement. Overall, our contributions in this work are that we:

- Present the concept of *numerical analytic provenance*, a novel approach to studying analytic provenance in visual analytic systems by visualizing the changes to their state as users interact via the proxy of changes to their underlying machine learning models.
- Provide a prototype tool to illustrate this concept that extracts models from user study software logs and creates an interactive spatialization with features to explore the analytic trails of users in detail.
- Evaluate our tool and this concept with two case studies on visual analytics systems.

2 Related Work

Analytic provenance in the visual analytics community broadly includes consideration for the history of how an analyst progressed through the various stages of his or her analytic process [35, 37, 20, 15, 36, 45]. Because visual analytics leverages human reasoning with a computational system, understanding how users build knowledge and insight can have implications for evaluating tools, as well as identifying ways to enhance collaboration between the user and the computer [36]. There are multiple stages to effectively analyzing user interaction histories to gain such an understanding about the user, including the most applicable to this work: encoding the interaction data and recovering semantic meaning behind the user's actions [35, 18].

The field of analytic provenance offers many examples of how to capture and encode this type of data [3, 13, 16, 21, 25, 31, 38]. One problem is that the desired level of granularity for understanding users' provenance is vastly different from the level at which standard computer software directly represents and logs interaction [21]. On one end, we seek to find patterns in semantic intentions of users, e.g. instances where two people may have a differently expressed high-level intention or strategy. On the other end, we have a wealth of recorded low-level system events like mouse movements and clicks.

One method to get semantic details from low level information is to carefully code the interaction data by hand [38]. In fact, it has been shown that process and strategy can be recovered this way [16], but the process is tedious and slow. Another solution is to build software with an organization scheme for interactions in mind. Systems taking this approach can provide powerful tools for users to examine their own analytic provenance trail in real time, and even organize it into useful, human-readable categories. Examples include VisTrails, HARVEST, CzSaw, and Graphical Histories [3, 11, 21, 30, 25], which capture sequences of states and visualize them for the user to use for navigation through the analytic process. For example, showing users a series of thumbnails of previous visualization states helps them recall aspects of their process and return to a previous state quickly if they decide to go back [25]. An additional layer of complexity can be added to show branching [39, 17]. However, the concepts in these works are built to work with specific software and while the concepts may generalize, the automation does not. There are a number of survey papers that provide a deeper set of examples from the broad spectrum of work in analytic provenance research (see [20, 14, 37]), but the central problem of gaining deep insight from low-level interactions remains a theme.

Recently, there are visual analytics tools to help with generic log data. Han et al. [23] process logs and present the user with an interface for organizing low-level tasks and building up higer-level ones. Zgraggen et al. [1] present a visual query language that can work over event sequences captured from logs to give a user the search capability empowered by regular expressions in text data. Chen et al. [12] provide a visual analytics system for sequence data that includes the use of the minimum description length (MDL) principle to help group interaction patterns automatically.

In this paper, we take a different approach to managing this challenge. We focus on the case where there is a software state that encodes the semantics of the user's sensemaking, and the state can be converted into a high-dimensional vector without direct human involvement per state. We call this approach *Numerical Analytic Provenance* and detail it in Section 3. Specifically, it is intended for the case when the system being studied uses machine learning as part of an interactive system,

and the changing, underlying machine learning models can be represented as a vector. The idea behind a numerical representation of the state of a visualization system was previously suggested by van Wijk [44] and adopted as the basis of the P-Set model by Jankun-Kelly et al. [27]. Our use of numeric analytic provenance extends these works and demonstrates how such an encoding can be applied to the visualization and analysis of users' interaction trails with visual analytics systems. The main savings is that the vectors can be created as the software runs or by processing log files with scripts as opposed to by hand.

To gain insight from this mathematical representation, we use visualization for the high-dimensional space, projecting the state vectors into a 2D space as dots (e.g. using Multidimensional Scaling (MDS) [32]). These dots are connected by lines to show sequences, as in connected scatterplots [24]. By using this compact representation, we have room to connect additional interaction information, e.g. annotating with what a user did that caused her to land at a given state. Our visualization is also similar to Time Curves [2] and the Dynamic Network approach [42]. By using computational models as software states and encoding them as vectors, we position them in space and show progression through that space over time with lines.

3 Numerical Analytic Provenance

There have been multiple approaches to analytic provenance, but one critical problem is that in seeking to understand how people use software, it becomes necessary to follow their trail through a wide array of possible interactions. With increasingly complex software, the task of capturing and analyzing exactly what a user has done in a way that can be efficiently understood is still a challenge. Previous work in analytic provenance has involved numerous methods for capturing the broad spectrum of interactions and a variety of encodings [31, 3, 16, 21, 38] to make it possible to analyze these interaction streams. While some work has sought to automatically encode and analyze interaction streams, most of the efforts have involved coding by hand for different types of interaction [16, 21]. A methods of automatically encoding and analyzing user interactions was proposed in Brown et al.'s work to learn models about users based on their interaction data [9], but the technique compares models of users, not their analytic provenance.

3.1 Vector Space of Models

Instead of manually coding user interactions into a human-readable format before beginning to build an understanding, we propose automatically encoding a numerical representation of changes to the internal state that the analytic software undergoes during the analysis process. We refer to this concept as *numerical analytic provenance*, and the encoded states as the *state models*. Deciding how to encode the state in a general way is an open problem as a solution would require solving the same problems that are left unsolved by other provenance systems, namely automatically extracting meaningful tasks and actions from low-level event data. We focus on systems that use machine learning back-ends to aid the analytic process and when possible, simply use the vector representation of the machine learning model as a state. We assume the interaction that causes the model to be updated is a significant action and the model update a significant change to the display, making these events good pivots for a visualization of the user's process. These states are also straightforward to extract from a log if they have been included, in contrast to actual user intent or high-level action. The technique presented in this paper visualizes the model sequences in the space of possible models by creating a visual layout such that more similar models are drawn nearer to each other. Because our representation consists of vectors, we can use high-dimensional data visualization techniques to calculate a projection, and the resulting visualization shows the interactions performed by different users in context of each other and in context of the broad spectrum of possible software states.

Figure 2 illustrates the concept of projecting three users' analytic trails from their high-dimensional vector representation down to a 2D visualization. Using this projection approach, it becomes immediately apparent when users' paths become close to each other, and when similar models pack together indicating an interesting region of the

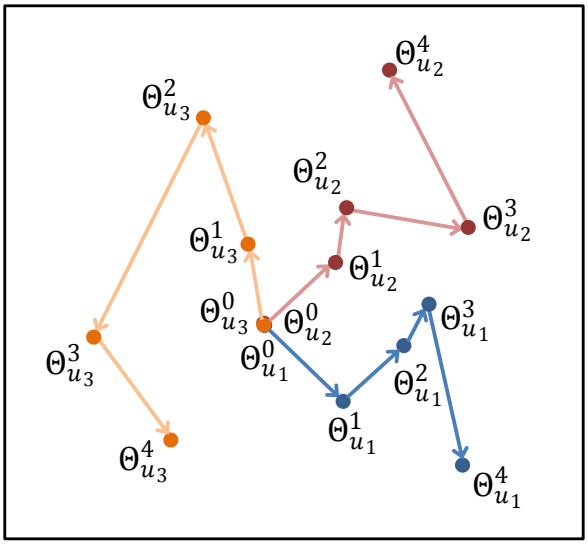


Fig. 2: This figure illustrates how the series of models created by a user's interaction trail can be represented by vectors and thus visualized for examination. Each dot represents a state model vector Θ^t_u that specifies the internal state of a system for one user, u, at one timestep, t.

overall state space. Further, this technique becomes more illuminating when we use the layout of the state models as a canvas to decorate with a wide array of other information. We provide context to the provenance by connecting the dots with lines that represent all the interactions that led to a state change, i.e. creating a connected scatterplot. Annotating the lines with these data integrates the interactions and their effects in one view.

Applying this technique for visualizing multiple users' numerical analytic provenance has a wide range of uses. By visualizing all the users' interaction histories together, we can compare their analytic processes to build an understanding of how and when they differ. For researchers or developers conducting experiments to evaluate analytic systems, this makes it possible to explore the trails of individual users and the relationships between their analytic processes. The analysis can reveal if there are areas of the model space that users always retreat from, or if different types of users pursue broadly different trajectories. For managers of multiple analysts, this not only allows oversight of progress, but has the potential to mitigate bias by alerting the manager when analysts are converging on one area of the model space. Additionally, if deployed as a provenance tool as part of a single user's interface (e.g. as in [11, 21, 25]), this technique could help the user understand not just what states she or he has seen, but also how they relate to each other.

3.2 Example Model States of Visual Analytic Systems

While any visualization can be represented by its internal state [44] to apply our proposed numeric analytic provenance approach, the use of a high-dimensional numeric vector to represent the state can have special implications for visual analytics systems. These systems often incorporate machine learning techniques or other data models to assist the user in exploring and analyzing data. Since machine learning and data models are mathematical in nature, they can often trivially be compactly represented as a high dimensional vector that can be used to represent the state of a user's analysis or exploration.

One type of visual analytics system that tightly couples a user's interactions with an underlying data model is *model-steering* visual analytics. These analytic systems capture user interactions with a data visualization and build a data model that encapsulates the changing data understanding of the user [18] . For example, ForceSPIRE [19] is an interactive visual tool for text analytics. The user is provided a visual layout of a set of documents and interacts with them via search,

moving documents relative to each other, and highlighting text. Based on these interactions, the system learns a model that characterizes the relative importance of the different words that appear in the documents. Each model update triggers a layout update and users iteratively refine the model through several interactive steps, leaving behind a trail of models about the words in the text corpus.

Conveniently, these models can also be considered state models, as they include the software state important to generating the visualization. By applying the numerical analytic provenance concept, we can visualize the relationships between the different data models the user constructed, each one showing the actual data features that were important to the user at the given time. We can annotate lines connecting these models with all the interactions between updates, indicating perhaps what documents were read and what words were highlighted. The process of exploring the analytic provenance is simplified, and the possibilities for discovery are broadened.

4 ModelSpace

In order to evaluate the numerical analytic provenance concept explicated by the previous section, we built a prototype interactive visual system, ModelSpace (see Figure 3), that enables analysis of user trails through the space of possible state models. In the following subsections, we describe the implementation and features of this prototype, beginning with the data required as input. We then describe the mechanism for computing a layout and the interactive tools that make analysis possible.

The ModelSpace prototype interface has been implemented for the web, using JavaScript with D3 [4], HTML, and CSS. The back-end software is responsible for processing log files, computing the projection, and serving the front-end with code and display data. It is implemented in Python 2.7 and uses the popular Numerical Python [43] and Scikit-Learn [10] packages for computation, and the Bottle [26] microframework for serving files.

4.1 Data for ModelSpace

Though the concept of numerical analytic provenance could be applied to a streaming context, with models updating the interface as they became available, our prototype is built to extract user interactions and model states from logs. For any given application, a function is needed that processes the logs to gather models and any accompanying information about user interactions between them. This can involve merging multiple records, e.g. log files from the software itself and digitized notes from an experimenter (as in the Doc-Function case study presented in a later section). The only absolute requirement is that the extracted models can each be represented as as vector, whether explicitly exported or constructed from the logs.

Both of our case studies produce internal models based on certain interactions, so our log processing simply extracts the times at which a model update was performed and captures the user input that caused the update and resulting model. The models created by these steps are represented by dots in the visualization in ModelSpace. The logs can include other actions performed between model updates, such as searching for words in documents in a text-analysis system. These non-model-generating interactions are all captured as they will be used in ModelSpace to annotate the lines that connect the dots, representing the actions taken between model updates.

When model changes can be reverted, i.e. with an *undo* feature, we keep track not only of the model update but the fact that it represents a reversion. The ability to backup in analysis is an effective tool for the user to expresses intention, informing us that the last model we saw could be a false step. ModelSpace represents this important contextual information with a curved line pointing back to the preceding dot.

Finally, it should be noted that while parsing a log file is sufficient for some applications, others will have more sophisticated data available and a more complex function for integrating it. In the Doc-Function case study below, for example, there were not only logs from the software itself, but notes from the experiment administrator about when each participant described certain insights. Information like this can be digitized with timestamps and merged at the time the logs are processed so that the visualization can reflect observations of participants along with the state models.

Overall, the prototype is designed to demonstrate the numerical analytic provenance concept specifically with two examples. While we made choices specific to those examples, we also sought to keep the visual representations and tools generic enough that they could be adapted to a wide range of data.

4.2 Calculating the Layout

When state models come to ModelSpace, they are vectors, generally in a high dimensional space that reflects the complexity of the analytic software. In order to visualize these high-dimensional states, we create a spatialization of these vectors that can be viewed in two dimensions. Since the desired view groups the states together based on their similarity, we use a Multidimensional Scaling (MDS), which is a type of projection of points into low-dimensional space (two-dimensions for our visual purpose) that optimizes for preserving the pairwise distances between points across the high- and low-dimensional spaces. This implies two useful features of the spatialization: first, similar models will be shown as dots that are close to each other, resulting in groups of similar models, and second, regions of the space of models will be reflected as regions in the projection. Other projections can achieve this result as well, but we chose MDS using Euclidean distance calculations because in comparison to other projections such as principal component analysis (PCA) [29] and t-distributed Stochastic Neighbor Embedding (t-SNE) [34] or alternative parameters to MDS, we found the results easiest to read. Note that any dimension reduction technique produces projection errors because generally high dimensional spaces inherently include information that cannot be represented with fewer dimensions. There are techniques to interpret errors (e.g., [6, 40]). Incorporating these techniques is out of the scope of this paper but will be an important future work for this project.

Calculating a spatialization of the states means that we can draw a scatterplot with a dot for each state, in which the more similar states are shown closer together. To show connections between states, i.e. those that occured in sequence in sequence for a single user, we connect the dots with lines. This results in a connected scatterplot. ModelSpace is flexible enough to incorporate other techniques of generating a connected scatterplot as long as the points in the plot represent the states of the system and states are connected based on the order in which the states were created. As described below, additional information can be added to this baseline visualization by mapping interaction data to lines and dots.

4.3 ModelSpace Prototype Features

Figure 3 shows ModelSpace, the interactive visualization we created for visualizing the type of numerical provenance data described in the section describing our approach, *Numerical Analytic Provenance*. We have designed the ModelSpace prototype to make possible an extensive analysis of interaction history data with a straightforward but powerful selection of tools. In the figure, the dots represent state models achieved by some participant at some point in the analysis task. The lines connect the models and represent order of the of the model updates. The arrows on the lines indicate the direction of progress from one state to the next. All the participants start with the same unweighted model in the example shown, so all the user lines begin at the same point. The layout of the models makes a clear comparison between the trails of different users and different user groups possible, but to find patterns in the models and interactions, some additional features are provided.

First, we make the rich interaction data available as annotations to the dots and lines, visible when the mouse cursor is over the element. In Figure 7, the orange rectangle is the mouse-over text for one dot, displaying the top ten most significant keywords that correspond to that model. In addition, the layout view supports panning and zooming. With a state model that includes human-readable features, as in the case of a model-steering system where the model features are dimensions of the original data, this provides insight into what was emphasized to the user at the point in their analysis corresponding to the model. When applicable, other information can be included here. For example, in one

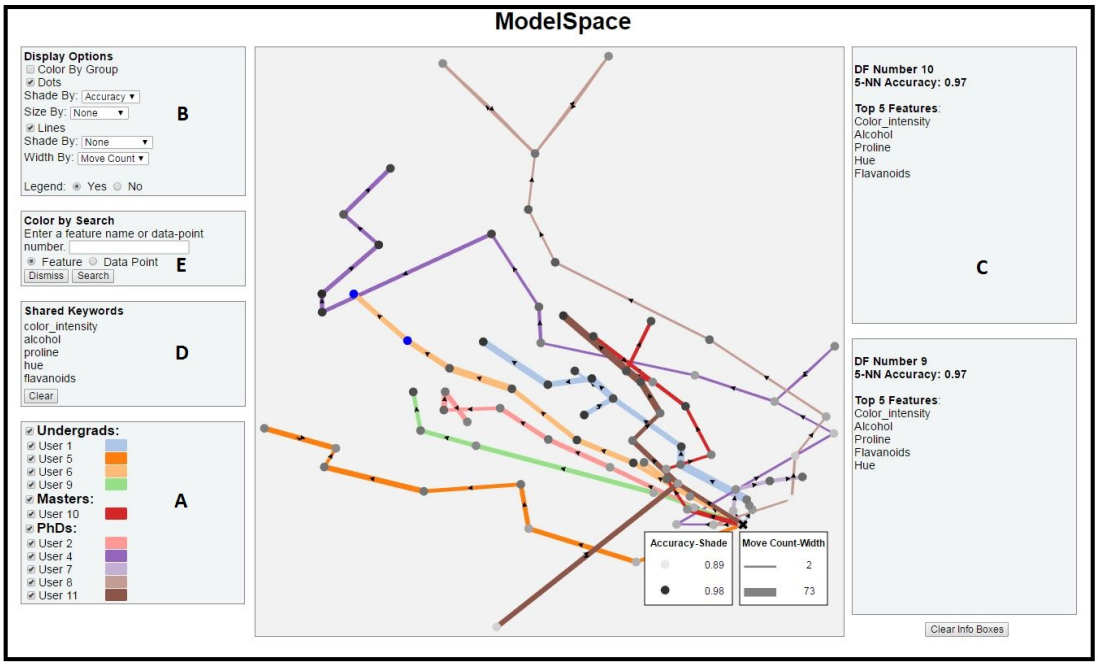


Fig. 3: ModelSpace, our prototype system for analyzing interaction trails. In this image we see a layout of all the models that have been created during the experiment described in the Dis-Function case study. Each model is represented by a dot, and we connect the dots for each user, representing the time between changes to the model. In this image, the width of the lines are varied by the number of points moved during the corresponding interaction and the dots are shaded by the accuracy values of the models. The legend in the bottom right of the visualization shows the move count and accuracy scores to which the line width and dot shadings respectively are mapped. In addition, three selected dots are highlighted in blue by a feature showing on the left panel which data features they have in common. The top five features of these models are also displayed in the two Info Boxes on the right.

of our case studies, the experiment used data with known ground-truth, so the accuracy of the user model relative to the ground truth can be shown here to show how similar this user's provenance had progressed toward some possible notion of optimum.

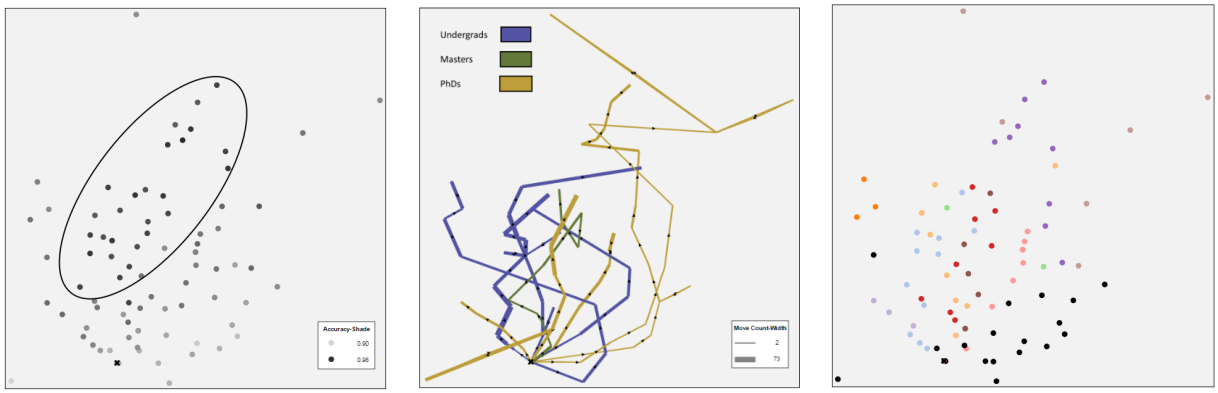
To make comparison between different users and user groups possible, there is a **User Selection Panel [A]** at the bottom of the screen. The check boxes enable the users lines and dots in the view, and the group selection boxes at the top of the panel toggle the entire group as a whole. These groups could be used to group the users by any helpful categories. The view can be further customized with the **Display Options Panel [B]**. First, the same user groups can be used to color the lines and dots with the *Color by Group* option, making comparison of group behavior much simpler. To simplify the view, dots or lines can be removed altogether if only one is needed. For example, in studying regions of the space by what the models have in common, the lines may be a distraction. This menu also controls mappings of data features to the display. Depending on the data available for the specific application, ModelSpace can map size and gray-scale shade of the dots and lines to data. For example, in Figure 3, the dots are shaded to the accuracy of the corresponding model. A legend is automatically added to the bottom to show the upper and lower bounds of the data mapping for whichever options are active.

When exploring the data, users will look at the information available as mouseover text for numerous models and lines. The mouseover modality alone makes it difficult to compare information. There are two **Info Boxes [C]** along the right side that persist the information associated with last two visual elements (dots or lines) to be clicked. The *Clear Info Boxes* button empties both boxes. When trying to compare the contents of multiple elements, seeing two alongside each other could be insufficient. The **Shared Keywords [D]** feature automatically detects what features different models have in common. The user can click on multiple dots, which are then colored blue to show they are being included in this comparison. The shared keywords box shows the keywords that the annotations for the selected dots have in common. For a model-steering system, i.e. in our case studies, the annotations of a dot include the names of the most important dimensions of the original data at that time. Therefore the shared keyword list shows the salient features of the data that are emphasized across the selected set of models. This feature can be used, for example, to discover what makes models that are shown close together actually similar to each other. Another usage would be to see what shared features were being shown to users at diverging points in their analysis.

Finally, there is a search feature, exercised by the **Color by Search [E]** box on the left side. The search accepts a string and highlights dots and lines that fulfill the query until *Dismiss* is clicked. This can be used to help find regions of interest or to look for elements that correspond to known entities in the analysis, as in Figure 8(c). For dots, this means highlighting models where the keyword was an important feature. Lines will be highlighted when the corresponding interaction sequences involved the search terms. For example, the user might have been reading lots of documents related to a certain word before updating the system about its importance. Searching for that word would show other times when users read such documents and when it was important to other models.

5 Case Studies

In this section, we demonstrate the capability of ModelSpace by using it to examine users' analytic trails from two different case studies. In both cases, the participants used a model-steering visual analytics system whose states can be easily converted to the high-dimensional vector representation used by the proposed ModelSpace approach. In these applications, as a user interacts with the system, the interactions are used by a machine-learning back-end to learn a new data model, which then updates the view so the user can iteratively improve it. These human-in-the-loop analytics systems are the most straightforward application of ModelSpace, because the user's interaction to create a new model represents a important point in the analytics, and that model's vector representation already exists.



(a) (b) (c)

Fig. 4: Views demonstrating features of the ModelSpace for Dis-Function. In (a), the dots are shaded by the accuracies of the models to which they correspond (higher accuracies are darker). The area marked by the ellipse contains the higher-accuracy models. In (b), the lines are colored according to group membership, and their widths encode the number of points involved in the corresponding model update. In (c), the dots representing models that emphasize *noise* features are colored black. The rest of the dots are colored based on the users to whom they correspond.

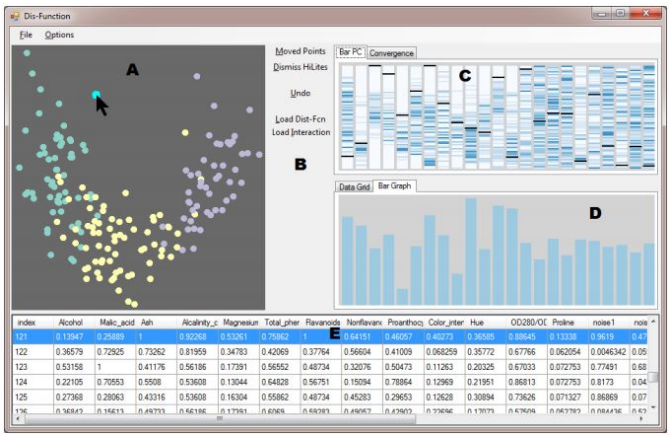


Fig. 5: The Dis-Function prototype. The user interacts directly with the visualization in (A) by moving the datapoints based on domain-knowledge. The options in (B) allow the user to undo a move and recalculate the layout after interaction. Based on how the points are moved, the underlying metric is updated, and through (C) and (D) the user is able to visualize the impact of the metric to the data. (E) displays the original data with the selected datapoint highlighted.

We cover each case study separately, first briefly describing the application and the experiment from which the data are collected, then explaining the mapping to ModelSpace and the application-specific features added. Finally, we discuss the insights gained by applying this technique.

5.1 Dis-Function

Dis-Function (Figure 5) is a prototype system that allows users to leverage their knowledge about data to build a machine learning model without having to understand the underlying algorithm. In this system, a user interacts directly with a visual representation of the data, specifically a two-dimensional layout of high-dimensional data.The layout is directly dependent on the model, so by providing feedback on the layout, the user (a data domain expert) causes the machine learning algorithm to update the model so that it is more consistent with the user's expectation. The newly regenerated model is used to create a new layout and the process can continue, iteratively improving the model, until the user is satisfied [8]. The model being learned at each step is a vector of weights, one for each data feature, and thus can be directly used in ModelSpace.

5.1.1 The Experiment

For this case study we used the experimental data from the authors of Dis-Function. Their participants include ten university engineering students (6 male, 4 female) at varying degree levels of study – five undergrads, one masters, and four Ph.D.s[1] The participants applied Dis-Function's model-steering technology to the Wine dataset from the UCI Machine Learning Repository [33]. These data have 178 instances, each representing one individual wine, and there are thirteen features, each representing a chemical component. The authors augmented the dataset with ten synthetic noise features (generated uniformly at random). Since the participants were not experts in the chemical composition of wine, Brown et al. provided them with labels that classified each data point as a certain type of wine by coloring the points in the display. The task, then, was to use Dis-Function by providing feedback to make the visualization more closely group the points with the same label, removing the influence of the noise.

5.1.2 The ModelSpace

As the experiment participants interacted with the system, Dis-Function logged all the interactions that produced model updates, and the updated models. These data were straightforward to extract and comprise a convenient set of state models for our application of ModelSpace. As seen in the bottom of Figure 3, each user's trail is represented by a different color. All the users start with the same initial model in the experiment, and thus all the user lines begin at the same place in ModelSpace.

Because the experiment with Dis-Function uses labeled data, we can actually calculate for each model produced by each user at each step, the accuracy of the model at predicting the given classes of the data. We apply the k-nearest-neighbor algorithm with $k = 3$ to make predictions with the Dis-Function models and use ten-fold cross validation to calculate accuracy scores. These accuracy scores are visible when the mouse cursor is over a dot, along with the names of the variables that had the highest contribution to the model at that point. When the mouse cursor is on top of a line, an annotation reveals which data points the user manipulated to cause the model update that happened during the period that the line represents. The same information used in the mouseover annotations can also be mapped to the color and size of the dots and lines, i.e. dots can be shaded or sized to reflect the accuracy of the corresponding model, and lines can be shaded or sized to reflect the number of manipulated data points.

[1]The user IDs shown in ModelSpace end at 11 but skip 3 due to one planned participant who was unable to participate.

5.1.3 Results

By exploring the ModelSpace generated for Dis-Function and interacting with its various features, we were able to capture some interesting trends. There is a clear indication that the higher accuracies are focused in one area of the visualization as seen in Figure 4(a), where the dots have been colored based on the model accuracy. The black ellipse shows the region with the strongest models. All the participants moved in directions of higher accuracy, but for some (labeled Users 5, 10, 11), the final model is not the most accurate one in the interaction trail. This can be seen in Figure 3 in which the dots are shaded with the accuracy values of the corresponding models. Following these users' lines from start to finish shows this non-monotonic progression. This outcome is not unexpected as the experiment participants were unable to see the accuracy values as they interacted with Dis-Function.

Another pattern is clear in Figure 4(b), in which each line's width is mapped to the number of points manipulated during that interaction period. Participants who travelled a shorter path overall, i.e., those who use fewer iterations to reach the final model, move more points during each iteration. Figure 4(b) also reveals another pattern with the point manipulations. Almost all the users are manipulating an increasing number of points as they are getting closer to the final model.

Applying the search feature, we can highlight all models that have the word "noise" in the name of one of their most salient variables. For these data, that variable name indicates one of the noise features added for the experiment. Figure 4(c) shows that in fact these noise features were diminished in importance after the first few interactions with the Dis-Function system. This helps showcase the effectiveness of Dis-Function at removing that artificial noise for its users.

Finally, we can investigate performance differences between groups of users. The participant group can be used to color the dots and lines. In Figure 4(b), we use this feature to see that the undergraduate and Ph.D. students' trails are moving mostly toward two separate directions. This suggests that these two groups are taking different approaches to interacting with Dis-Function.

Through these results we are able to gain a better understanding of the participants in the Dis-Function experiment, and the behaviours associated with different groups. ModelSpace also clearly underscores that through interactions with Dis-Function, all the users were able to improve their models to attain higher accuracies and reduce the significance of the noise features. While this was known from the publication about Dis-Function, the authors of that paper were not able to do such in-depth analysis about the patterns of interactions that lead to these results.

5.2 Doc-Function

Doc-Function [7] (Figure 6) is a visual analytic tool designed to enable sensemaking of text corpora through manipulation of a keyword spatialization. The spatial layout of the keywords encodes the similarity between keywords with respect to the documents in which they co-occur. Doc-Function allows users to manipulate the spatialization of keywords extracted from documents to perform model-steering without having to understand how the new model is generated. Based on the user's evolving understanding and knowledge of the documents in the corpus, the user can move the words relative to each other to reflect the correct relationships and groupings. Making changes causes an update to a model that reflects the relative importance of the documents in the corpus, triggering creation of a new corresponding spatial layout. In this way, Doc-Function is similar to Dis-Function, except it is designed for text. There are a number of differences in the technology, but with respect to ModelSpace, the main difference is that Doc-Function, taking advantage of the properties of text data, supports a wider variety of interactions and thus exports richer logs.

5.2.1 The Experiment

In order to visualize the numerical analytic provenance of the users of this tool, we obtained data from an experiment that was run to evaluate Doc-Function. The Doc-Function authors conducted an experiment with 13 participants at a national laboratory (name withheld for anonymous review) from four different job categories: professional analysts

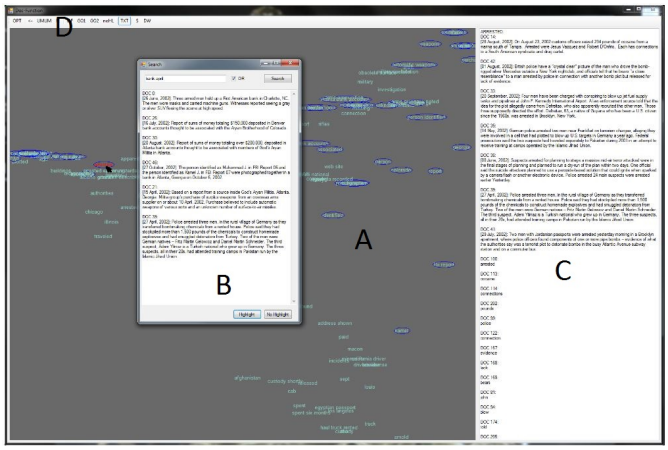


Fig. 6: The Doc-Function prototype. The user interacts with a projection of keywords (A) by moving them around into a spatialization that better represents his understanding of the similarity between the words. These interactions cause changes to a machine learning back-end. The pop-up window (B) allows the user to search for a list of documents that contain one or more words and right column (C) displays the documents that contain a particular word upon mouse-over. The buttons on the top (D) allow the user to perform actions like undoing a move and highlighting all the keywords that belong to a document. These features assist the user make more informed movements of the keywords.

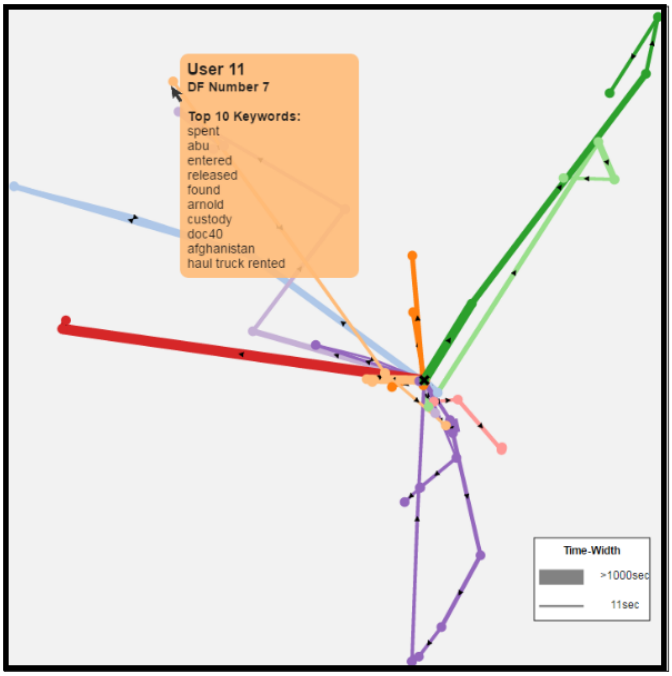


Fig. 7: ModelSpace of the Doc-Function. In this view, the widths of the lines encode the time spent by the user during that interaction. Additionally, two points have been selected (marked in blue) and their shared keywords are displayed in the *Shared Keywords* box on the left panel.

(2), scientists and engineers (5), interns (5), and administrative staff (1). The experiment used a data set designed for intelligence training. The 49 documents of the corpus contain a fictitious terrorist threat that each experiment participant was tasked to discover. Participants were encouraged to discuss their process in this think-aloud study, and given as much time as desired (typically just under an hour). Full details of the study are available elsewhere [7].

5.2.2 The ModelSpace

Just as with Dis-Function, we constructed a ModelSpace for the Doc-Function experiment by extracting the various data models about documents generated by the users with their interactions. Figure 7 shows the ModelSpace built for Doc-Function. The dots represent the models and the lines are annotated with information about the interactions that resulted in the models. All these elements are colored by default to show which user they represent.

The Doc-Function system has a richer interaction set than Dis-Function, and the logs reflect this diversity. The data include records of viewing documents, using the text search feature, performing model updates, and using the undo and reset functions. We loaded these interactions into ModelSpace as annotations to the lines and thus, rather than showing only model updates, we are able to show all types of interactions by participants that led them to model updates. In addition, we took advantage of the detailed notes from the experiment administrator by digitizing them to sets of observations with timestamps, and incorporating them into the annotations as well. The annotations include not only what interactions the user performed, but a distilled version of their think-aloud commentary about their insights and process.

Beyond annotating the visual elements with the collected data, we enable the other features to use this information as well. The search feature can be applied to both the set of salient data features associated with the dots, and the full set of information annotating the lines, as in Figure 8(a). The shade and thickness of the lines can also be mapped to the number of documents read, the number of word searches made, time spent during the interactions, and the number of words moved. This rich set of available information is simple to incorporate in ModelSpace, and makes it possible to explore the analytic provenance of the participants in much deeper detail.

5.2.3 Results

Just as with Dis-Function, the ModelSpace of Doc-Function reveals a number of interesting insights. By using the ModelSpace search feature to highlight dots and lines that contained keywords, we saw that the words such as "Aryan" were nearly ubiquitous across the models and interactions (see Figure 8(a)). We examined the differences between our participant groups, as seen in Figure 8(b), by mapping color to group identity (rather than individual user). It becomes visually apparent that the interns (colored in red) moved in a direction with their model building that diverged from other participant groups. This figure also shows that the interns stand out as using lower overall numbers of searches. In fact, we can see by switching options in the left panel, that interns moved fewer keywords in their model-updates, and read fewer documents[2].

Looking in more detail at how many documents the participants read, we find that there is a trend of having higher read counts in the beginning and lower read counts as the users approach their final models. This can be seen in in Figure 8(c), where the lines are shaded based on the read count. The starting lines for each participant appear darker than the lines before the final model.

Through ModelSpace, we were able to gain some interesting insights about the users' behavior and their approach to the analysis task. This would have been difficult to accomplish with manual inspection of logs and interaction trails.

[2]We cannot guarantee the documents were read in full, but this indicates the user was able to see their content.

6 Discussion

In the process of making and using the ModelSpace tool, we have investigated several possibilities and revealed areas for future work that we will discuss in this section. First, we discuss other uses for ModelSpace beyond analysis of experimental data. Next we describe application areas beyond model-steering analytics, including some preliminary results with interaction data from a visual search task. We then discuss the complexities of the step of projecting the set of models in more detail. Finally, we discuss future directions and implications of this work.

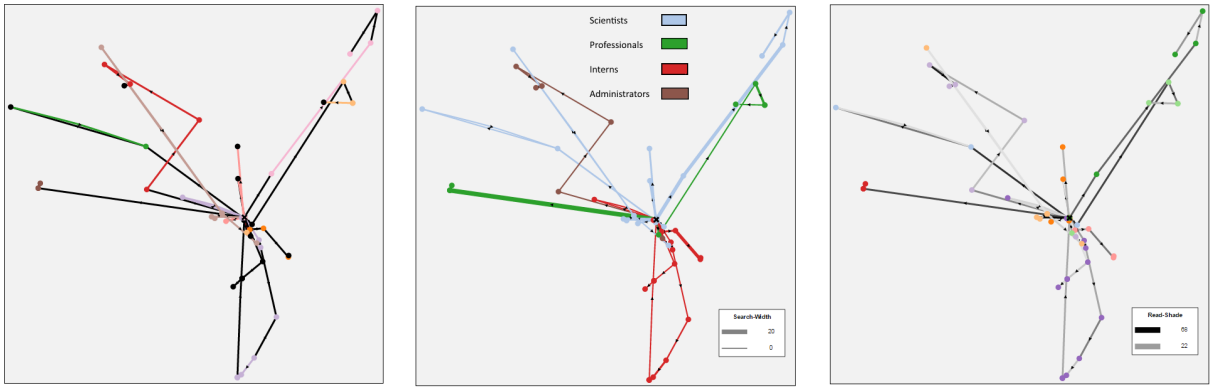
6.1 Uses of Numerical Analytic Provenance

ModelSpace can be used in other applications besides model-steering visual analytics systems, because the requirement for the input is simply that there be some software state that can be extracted and converted to a vector. For people tasked with understanding how people use software, this broad applicability is an exciting prospect because existing methods for analyzing the results of experiments evaluating software can be cumbersome. Aside from analyzing the results of experiments, this technique could be more broadly applied to help users understand their own analytic provenance. Similarly to the usefulness of undo history in a web browser or more sophisticated analogs in previous analytics research [21, 3, 25], this technology could be used to show users not only their interaction history, but a visualization of their trails with context and ability to move back and forth between their most useful state models. Visualizing for the user the space of states created during their work could be a transformational way to make this helpful general mechanism stronger.

Even as the visualization of this space could be useful to individual users, it could also help managers overseeing multiple analysts. Someone responsible for the efforts of a team trying to find a threat in a massive corpus of text data could use a ModelSpace-like tool to view the ongoing progress of analysts and make sure they were covering different areas of the possible model space, helping to mitigate bias, which is a subtle and difficult problem facing such efforts today. Another relevant domain from interactive machine learning is recommender systems. When evaluating the quality of a recommender system, researchers use statistical measures. But with a tool like ModelSpace, it would be possible to gain a deeper understanding of how different users implicitly and iteratively create models of what they like through their ratings, comments, purchases, and other interactions.

6.2 Application Areas

Though the case studies were both performed with model-steering visual analytics systems, we believe this technology lends itself to a wider array of applications. As one test of alternative applications, we applied ModelSpace to a collection of data from an experiment with an image search task. The "Finding Waldo" study by Brown et al. [9] included collecting data about how a set of participants found a drawing of a certain person in a large hand-drawn image using a search tool that provided basic navigation controls. In that work, the authors created multiple encodings of interaction sequences collected from a study of users performing the search task. They showed that the users could be distinguished into groups by performance and other factors by applying machine learning. We apply ModelSpace to the *state space* encoding of that work, which characterizes a participant's interactions up to time t by the sum of all states of the software they have encountered by that time. States in this case are the states of the visual search window and thus encode the zoom level and where the user's view is centered. ModelSpace can use these state vectors directly, and we visualize the projection in Figure 9. Because there is no contextual interaction information available from this application, we include these results only in the Discussion as a way of demonstrating the wider applicability of the technique. In the figure, the groups of participants with the fastest and slowest completion times are highlighted by color, and we can see how different their trails are through the space of models at a glance. The faster users have covered a broader area of the model space than the slower users. Further, this application showcases a much larger



(a) (b) (c)

Fig. 8: Views demonstrating various features of the ModelSpace for Doc-Function. In (a), some lines and dots are colored black to indicate the corresponding interactions and models reference the word "Aryan". In (b), the lines are colored by the user groups and the widths of the lines are mapped to the number of searches made. Note that the "Interns", colored in red, show an analysis trajectory that is distinctively different from the others. In (c), the lines are shaded to reflect how many documents were read and the dots are colored for the individual users.

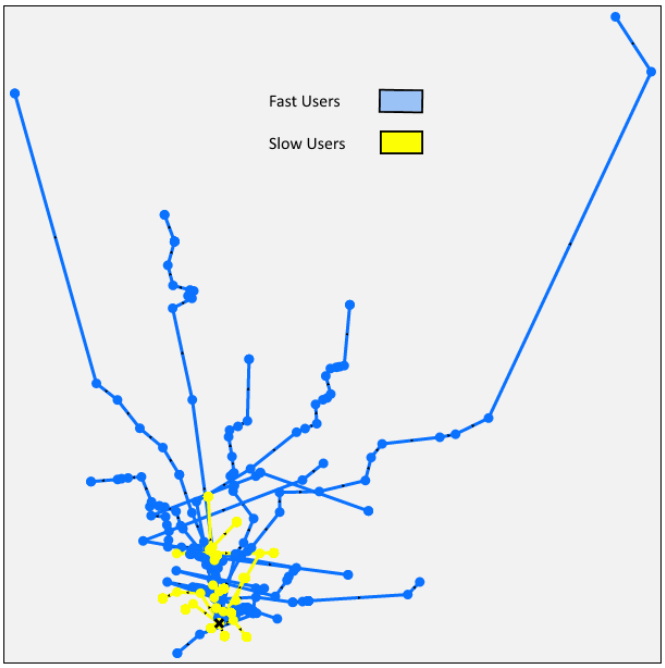


Fig. 9: ModelSpace of the directional vectors of users after interacting with a state space system to locate the drawing of a certain person. The blue lines represent the fast group of users and the yellow lines represent the slow group of users.

sample of states, showing how the compactness of representation is useful as experiment sizes grow.

6.3 Future Work

In this section we provide a number of suggestions to future users of this technology. First, we believe it is possible that a projection method outside the scope of this work could be a better fit to this type of data. To our knowledge, there is no criteria for selecting a projection that can automatically detect which will be the most useful for this type of analysis, so a feature to let the user decide may be the best approach.

The current version of ModelSpace makes it possible to review a significant amount of information, but statistical and model-building tools within ModelSpace could take the analysis to the next level of detail. For example, after discovering a pattern in the main visualization, e.g. a connection between the number of interactions performed before generating a model and the model's likelihood of including some particular variable, there could be a feature to test the hypothesis by calculating a correlation between those occurrences in the data. Perhaps, after discovering interesting comparisons between two groups of models, the user could indicate the groups, and the system would respond with an automatic categorization of what model features differentiate them. Finally, with analytics software getting increasingly complex, there may be a need to examine more sophisticated types of models. For example, a recent model-steering innovation [5] allows multiple models to be considered at once. The ModelSpace concept could be extended to accommodate these increasing complexities.

We believe this numerical form of analytic provenance opens up new avenues for using visualization to explore users' interaction patterns. Unlike traditional visualizations of interaction logs, the use of ModelSpace allows immediate comparison of the analysis trails between multiple participants. A thorough examination of the benefits and limits of this approach will require others to apply it to their own problems and evaluate it for their purposes, and we look forward to seeing the results of such applications.

7 Conclusion

In this paper, we have discussed a novel approach to analyzing user interaction trails with interactive machine learning systems. Numerical analytic provenance makes it possible to study analytic provenance by constructing state models from the logs of interactive systems, particularly when certain crucial interactions produce a new model. Vectors representing models are shown in a layout that reflects their similarity, creating a backdrop for an interactive visualization annotated with the full spectrum of available interaction information. To showcase this concept, we have provided an implementation, ModelSpace, with an array of features for exploring analytic provenance in a visualization of the state models. We applied ModelSpace to two case studies of model-steering visual analytics systems using logs generated from their original evaluation experiments. Additionally, we provided an example application of ModelSpace to a non-visual analytic system, showing how to apply the vectorization principle for interactions with an image search tool. The case studies demonstrated the effectiveness and wide applicability of ModelSpace and of numerical analytic provenance concept by making it possible to explore the interaction data from those experiments and reveal patterns that would have been difficult to discover without such a tool.

References

[1] *(s|qu)eries: Visual Regular Expressions for Querying and Exploring Event Sequences*. ACM âĂŞ Association for Computing Machinery, April 2015.

[2] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. Dragicevic. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE transactions on visualization and computer graphics*, 22(1):559–568, 2016.

[3] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *Visualization, 2005. VIS 05. IEEE*, pp. 135–142. IEEE, 2005.

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[5] L. Bradel, C. North, and L. House. Multi-model semantic interaction for text analytics. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pp. 163–172. IEEE, 2014.

[6] M. Brehmer, M. Sedlmair, S. Ingram, and T. Munzner. Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*, pp. 1–8. ACM, 2014.

[7] E. T. Brown. *Learning from Users' Interactions with Visual Analytics Systems*. PhD thesis, TUFTS UNIVERSITY, 2015.

[8] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 83–92. IEEE, 2012.

[9] E. T. Brown, A. Ottley, H. Zhao, Q. Lin, R. Souvenir, A. Endert, and R. Chang. Finding waldo: Learning about users from their interactions. 2014.

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[11] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pp. 745–747. ACM, 2006.

[12] Y. Chen, P. Xu, and L. Ren. Sequence synopsis: Optimize visual summary of temporal event data. *IEEE transactions on visualization and computer graphics*, 24(1):45–55, 2018.

[13] P. Cowley, L. Nowell, and J. Scholtz. Glass box: An instrumented infrastructure for supporting human interaction with information. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pp. 296c–296c. IEEE, 2005.

[14] S. M. S. da Cruz, M. L. M. Campos, and M. Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *Services-I, 2009 World Conference on*, pp. 259–266. IEEE, 2009.

[15] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1345–1350. ACM, 2008.

[16] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang. Recovering reasoning processes from user interactions. *IEEE Computer Graphics and Applications*, (3):52–61, 2009.

[17] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1663–1672. ACM, 2012.

[18] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 473–482. ACM, 2012.

[19] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 473–482. ACM, 2012.

[20] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[21] D. Gotz and M. X. Zhou. Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.

[22] T. Grossman, J. Matejka, and G. Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pp. 143–152. ACM, 2010.

[23] Y. Han, G. D. Abowd, and J. Stasko. Flexible organization, exploration, and analysis of visualization application interaction events using visual analytics.

[24] S. Haroz, R. Kosara, and S. L. Franconeri. The connected scatterplot for presenting paired time series. *IEEE transactions on visualization and computer graphics*, 22(9):2174–2186, 2016.

[25] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1189–1196, 2008.

[26] M. Hellkamp. Python bottle. `http://bottlepy.org/`, 2016 (accessed September 21, 2016).

[27] T. Jankun-Kelly, K.-L. Ma, and M. Gertz. A model and framework for visualization exploration. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):357–369, 2007.

[28] D. H. Jeong, W. Dou, H. R. Lipford, F. Stukes, R. Chang, and W. Ribarsky. Evaluating the relationship between user interaction and financial visual analysis. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, pp. 83–90. IEEE, 2008.

[29] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[30] N. Kadivar, V. Chen, D. Dunsmuir, E. Lee, C. Qian, J. Dill, C. Shaw, and R. Woodbury. Capturing and supporting the analysis process. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pp. 131–138. IEEE, 2009.

[31] M. Kreuseler, T. Nocke, and H. Schumann. A history mechanism for visual data mining. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 49–56. IEEE, 2004.

[32] J. B. Kruskal and M. Wish. *Multidimensional scaling*, vol. 11. Sage, 1978.

[33] M. Lichman. UCI machine learning repository, 2013.

[34] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[35] C. North, R. Chang, A. Endert, W. Dou, R. May, B. Pike, and G. Fink. Analytic provenance: process+ interaction+ insight. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pp. 33–36. ACM, 2011.

[36] W. A. Pike, J. Stasko, R. Chang, and T. A. O'Connell. The science of interaction. *Information Visualization*, 8(4):263–274, 2009.

[37] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics*, 22(1):31–40, 2016.

[38] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1237–1246. ACM, 2008.

[39] A. Singh, L. Bradel, A. Endert, R. Kincaid, C. Andrews, and C. North. Supporting the cyber analytic process using visual history on large displays. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, p. 3. ACM, 2011.

[40] J. Stahnke, M. Dörk, B. Müller, and A. Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE transactions on visualization and computer graphics*, 22(1):629–638, 2016.

[41] J. J. Thomas and K. A. Cook. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005.

[42] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE transactions on visualization and computer graphics*, 22(1):1–10, 2016.

[43] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[44] J. J. Van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pp. 79–86. IEEE, 2005.

[45] K. Xu, S. Attfield, T. Jankun-Kelly, A. Wheat, P. H. Nguyen, and N. Selvaraj. Analytic provenance for sensemaking: A research agenda. *IEEE computer graphics and applications*, 35(3):56–64, 2015.