Chive: A System for Collaborative Exploratory Data Analysis

Chuan Yu Foo
Stanford University
cyfoo@stanford.edu

ABSTRACT
Exploratory data analysis is a process which involves rapid iteration and creation of multiple views of a dataset to explore the data, test hypotheses and find interesting patterns. To facilitate this process, we present Chive, a system for collaborative exploratory data analysis which keeps track of the complete history of analysts’ exploratory process, allows them to quickly revert to any visualisation produced in the process, to search through this history for related visualisations, and to collaborate by sharing their exploration histories with other analysts and building on their work.

Author Keywords
Exploratory data analysis; History; Collaboration;

INTRODUCTION
One of the ways in which interesting patterns in data are found is through exploratory data analysis. In exploratory data analysis, the analyst starts with a dataset and explores different views of the data produced by transforming the data (e.g. by filtering, aggregation or creation of derived data) and encoding the data in different ways (e.g. as a scatter plot, a bar chart, and so on).

Throughout this process, the analyst may generate a number of different hypotheses and in the process of exploring these hypotheses, construct a number of different but related visualisations. As a result, analysts often generate a large number of intermediate visualisations, which, though they may be uninteresting in their own right, serve as a record of the analyst’s progress and findings, as well as starting points for further exploration by the analyst.

However, most existing tools for exploratory data analysis (in particular, Tableau) lack functionality which supports this rapid iteration between multiple intermediate visualisations – in existing tools, analysts must manually save each visualisation they wish to preserve as a separate document. This method is cumbersome, discouraging the preservation of intermediate visualisations, and it also fails to capture the sequence in which the visualisations were created, which can be helpful in understanding the analyst’s thought process.

The goal of Chive is to address this problem by creating a system which unobtrusively keeps track of the complete history of an analyst’s exploratory data analysis without requiring additional effort by the analyst. Further, once a complete history is available, a natural extension is to support collaborative exploratory data analysis, in which multiple analysts can share and exchange their exploration histories, enabling the more efficient exploration of datasets by allowing them to see what hypotheses have already been considered, and to focus on areas which have yet to be explored.

RELATED WORK
The problem of keeping track of the history of an analyst’s exploratory data analysis has been previously addressed by Heer et al. [1]. Heer et al. developed an extension to Tableau which keeps track of high-level actions (such as shelf, data and analysis actions) performed by the analyst in a history tree which also captures the sequence in which the actions were performed and the intermediate visualisations produced. In addition, the extension also supports adding metadata such as titles and descriptions to the intermediate visualisations, and searching and filtering on these visualisations.

Various authors have also addressed the topic of collaborative visual analysis. Two notable examples are the Many Eyes visualisation site [3], and a system built by Heer et al. [2]. In both cases, a number of different pre-created interactive visualisations of pre-selected datasets are made available, and users can interact with these visualisations to explore different views of the data (within the constraints of the visualisations provided), which they can share with other users, annotate and comment on. However, at present, we know of no examples in which users can start with an arbitrary dataset of their choice and collaboratively design a visualisation or explore different views of the data.

METHODS
Data Analysis Functionality
As we were unable to find an exploratory data analysis system which we could extend or build on, we built a minimal clone of Tableau. As reproducing the full functionality of an exploratory data analysis system was not the focus of this
project, only a small subset of key functionality was imple-
mented in Chive.

In particular, the main features implemented include: import-
ing data from arbitrary CSV files including automatic detec-
tion of quantitative and nominal variables; the visualisation of any subset of variables in scatter plots; filtering over quan-
titative or nominal variables; and the creation of derived vari-
ables. Due to time constraints, features such as data aggrega-
tion and different encodings (such as mapping variables to size, colour, etc) were not implemented.

Exploration History
At a high-level, the three main features we wanted the explo-
ration history to have were:

(i) The ability to keep track of every action taken by the
analyst, as well as all intermediate visualisations pro-
duced, without requiring any intervention by the analyst

(ii) The ability for analysts to add titles and descriptions
to the intermediate visualisations to detail their thought
process

(iii) The ability for analysts to easily navigate and search
through the history to find visualisations of interest

(iv) The ability for analysts to share and exchange histories
and to build on each other’s work

The exploration history subsystem was designed with these
goals in mind.

Internal Representation
Internally, the history is maintained in a history tree con-
structed as follows. On the creation of a new document, a
root node is created to represent the loading of the data. At
every point in time, the current state of the visualisation (and
the analyst’s process of exploration) corresponds to a node in
this tree.

When the analyst performs an action (such as adding or re-
moving a variable from a shelf, filtering data, and so on), the
action is recorded, along with any data necessary for undoing
the action. A new node is generated in the tree corresponding
to this action and the resulting visualisation, and its parent is
set to the node corresponding to the previous state of the visu-
alisation. The current state of the visualisation is then updated
to reflect the change.

Undo-ing an action corresponds to reverting the visualisation
to the state corresponding to the parent of the current node in
the tree, and is accomplished by making the changes neces-

sary to undo the action. In general, the visualisation may be
reverted to any state in the tree by following the path from
the current state to the target state and making the appropriate
changes to the visualisation.

To facilitate searching of history, for each intermediate visu-
alisation, a hash of the data and variables visualised is com-
puted and stored in an index. This makes it easy to search for
visualisations with views similar to other visualisations.

Interactive Views
As screen real estate is limited, it is infeasible to display the
full history in any form at all times within the application.
Furthermore, a complete view of the full history would prob-
ably not be useful to analysts during most of their analysis.
As such, a “local view” of the history near the current state
was included in the frame of the main application window,
along with a “popout button” which when pressed, opens a
new window with a “global view” of the history. The two
views are described in more detail below.

Local History View
Informal observation and experience suggested that interac-
tions with “local history”, that is, states close to the current
state in the history tree, to be infrequent, with the excep-
tion of undo and redo. On the other hand, when viewing a
visualisation that has many other visualisations based on it
(equivalently, whose corresponding node in the history tree
has multiple children), it may be helpful for an analyst to get
an overview of nearby states to understand the hypotheses
explored by other analysts so that he/she can avoid revisiting
previously explored hypotheses.

As such, in the local history view (shown in figure 1), we
decided to display the current node and its siblings, as well
as the parent node. In addition, in order to allow analysts to
inspect the children of the nodes in the limited space avail-
able, a “details-on-demand” approach was adopted in which
the children of a node are displayed when the mouse hovers
over the node. Further, to facilitate the viewing of titles,
descriptions and other metadata relating to the associated vi-
sualisations, a tooltip containing these details is also shown
when the mouse hovers over a node. Finally, analysts can re-
vert to any visualisation represented by the nodes by clicking
on the corresponding node; they may also edit the metadata
of any node by right-clicking on the corresponding node.

Figure 1. An illustration showing the local history view. (i) The “popout
button” which brings up the global history window (ii) A tooltip showing
the details of the node the mouse is currently over (iii) The current node
is highlighted in yellow (iv) The children of the node the mouse is over
are shown in the box. These nodes can be interacted with like any of the
other nodes.

As is natural, the local history view is updated as the analyst
makes changes to the visualisation. However, to avoid dis-
tracting the analyst, no transitions are used when the view is
updated due to the analyst making changes. This is consist-
tent with informal observations that suggest that throughout
the exploration process, analysts rarely attend to the local history view for information about their changes.

On the other hand, changes to the local history view caused by directly interacting with view by clicking on a node to revert the visualisation to that state do include staged transitions to help the analyst understand the effect of the change. For instance, clicking on the parent node results in the staged transition illustrated in figure 2.

**Global History View**

In contrast to the local history view, the global history view aims to allow analysts to trace the full history of their exploration process to understand the different sequences of visualisations explored, to search for other interesting or relevant visualisations by themselves or others, and to quickly move between different visualisations.

The biggest consideration in designing the global history view (shown in figure 3) was finding a way to allow analysts to visualise and explore large exploration histories that can potentially contain hundreds of intermediate visualisations in multiple different branches.

Given that most analysts are familiar with the tree structure of the filesystem, the same metaphor was used for displaying the exploration history. This encoding was further supported by the fact that there is a strong correspondence between the view this generates and the view analysts might have when saving their intermediate visualisations in the filesystem.

![Image](54x276 to 297x409)

**Figure 3.** An illustration showing the global history view. (i) Each node has a bar corresponding to it, showing, from left-to-right, a control to expand/contract its children; the total number of descendants (in black); the total number of descendants with titles (in red); the title of the node (ii) A tooltip showing the details of the node the mouse is currently over (iii) The current node in the main window is highlighted in yellow (iv) Search results are shown here for a search for nodes with titles – the current node is also highlighted in yellow here.

To capitalise on the correspondence between the two metaphors, the global history visualisation was designed to be as similar to a typical filesystem browser as possible. Exactly as in a filesystem browser, for each node, there is a rectangle, indented to reflect parent-child relationships. For nodes with children, there are expand/contract controls as in a file browser.

Each rectangle contains, in addition to the title of the node (if present), counts of the number of descendants and number of descendants with titles. Since the branches all contracted by default, this helps analysts quickly decide which branches might be worth looking further into (those with many descendants with titles), and which branches are probably dead ends (those with few descendants or few descendants with titles).

As in the local history view, hovering over a node brings up a tooltip showing metadata associated with the node, clicking on a node reverts the visualisation to the state represented by that node, and right-clicking on a node allows the analyst to edit the metadata associated with the node.

To allow analysts to browse the full history and see the visualisation represented by the node they are looking at simultaneously, the global history view is linked to the main application in the sense that changes made to one are immediately reflected in the other. For instance, when the global history window is opened by clicking on the button in the main window, the global history view expands all the parents of the current node and scrolls to center on the current node. Similarly, clicking on a node in the global history view reverts the visualisation in the main window to that corresponding to the node clicked.

**Search**

In the global history view, analysts are able to search for visualisations in the history on criteria such as title, description and username.

In addition to these criteria, analysts are also able to search for visualisations which show the same set of variables as the current visualisation, or which show the same data (after filtering and transformation) as the current visualisation. This allows analysts to quickly find other related visualisations to understand what hypotheses they or other analysts have explored in the past.

In line with the filesystem metaphor, the search feature mirrors that of typical file browsers quite closely – search results are shown in a table, with rows that can be clicked to revert the visualisation to the corresponding state, allowing analysts to rapidly browse through and view the visualisations found.

**Collaboration**

To allow analysts to exchange history and build on each other’s work, Chive includes the ability to export and import exploration histories for the same dataset.

Furthermore, to help analysts avoid exploring hypotheses already explored by other analysts, a notification is displayed whenever an analyst creates a visualisation that displays the same data as other visualisations in the exploration history. Clicking on this notification brings up the full history window with search results showing the other visualisations, allowing the analyst to quickly browse through previous visualisations to avoid replicating previous work.

**RESULTS**

**Evaluation**

To verify the assumptions made in designing Chive, as well as to evaluate the effectiveness of Chive for exploratory data analysis, we conducted an informal user study. Due to time
Figure 2. An illustration showing the staged transition on clicking a parent node in the history view. From left to right: the analyst clicks on the parent node to revert the visualisation to that state; the new current node is highlighted; the diagram shifts downward; the child nodes are hidden; the new current node moves to its position as a child of its parent; the new parent node and its other children appear; the children of the new current node are shown by default.

constraints, we could only survey 2 participants for this study. The participants were both undergraduate computer science majors at Stanford.

To verify the assumptions made, participants were given the opportunity to use the tool freely to generate visualisations for a given dataset (specifically, the movies dataset from the class) for 5 minutes. The actions they performed using the history controls were observed and recorded. The main question we wanted to answer for this part of the study was how, when and why do participants use local and global history.

In the 5 minutes of exploration, it was observed that both participants rarely interacted with either the local or global history controls. While both participants initially used the local history control for undoing actions, they switched to using the keyboard shortcut CTRL+Z when informed that the shortcut was available. The predominance of undo as the main interaction with history is consistent with Heer et al. [1], who found the same result.

When questioned about their use of local history, both participants mentioned that the tree view showing the current node, its parent and its siblings, was not very useful – both mentioned that they would prefer some way to view the entire history tree with zooming functions.

Furthermore, both participants found the spatial layout of the children in the details-on-demand view confusing – when hovering on different nodes, the children are displayed in the same region of space, which they felt was confusing. Indeed, as there are no transitions when switching between the children of different nodes, the expectation of object constancy on the part of the participants was reasonable. This suggests that in future work, transitions could be added to minimise confusion.

With regard to the global history, one participant did not like the filesystem metaphor, expressing a preference for a tree view similar to that in the local history. The other participant was neutral about the filesystem metaphor. However, both participants remarked that locating the parent of nodes became difficult in the global history when each of the siblings have multiple children. These two remarks suggest that perhaps analysts might prefer to have parent-child relationships and the branching of the history nodes foregrounded, which would be accomplished by switching to a tree view as suggested by the first participant.

Both participants suggested that instead of displaying branch statistics (such as the count of number of children or number of nodes with titles), the tree view could hide “uninteresting” nodes and only display “interesting” nodes. This idea was considered during prototyping, however, one issue is that it might be difficult to show parent-child and branching relationships while hiding “uninteresting” nodes.

The search feature was well-received by both participants, who found it easy to use. However, one participant noted that it might be useful to be able to search for visualisations which visualize a superset of the given variables rather than visualisations which visualize exactly the given variables. Furthermore, the same participant suggested that searching for matching views should also find views that are similar, in that views which differ by a small number of data points should also be included in the search results. Both suggestions are interesting to consider as directions for future work.

DISCUSSION

Chive was designed to support rapid and collaborative exploratory data analysis by allowing analysts to focus on exploring hypotheses and building visualisations without worrying about having to manually keep track of the exploratory process and the intermediate visualisations produced. Due to the lack of time, the effectiveness of Chive for collaboration was not evaluated. However, preliminary user testing suggested that users found the unobtrusive tracking of history and the information about related visualisations useful. User feedback also suggested that the visualisations and interactions of history both local and global could be better tailored to suit users’ needs.

FUTURE WORK

A significant weakness with Chive is that the manual export, import and exchange of history files is tedious and makes collaboration difficult. Future work could explore the possibility of real-time collaboration on the web with the exchange and sharing of history managed by a web application.

Reframing the preservation of exploration history as version control for data visualisations opens up a large number of pos-
sibilities for future improvement to the history system, including history operations such as merging histories, comparing histories, and replaying parts of histories from one branch on another branch.

IMPLEMENTATION
Chive was implemented using the following libraries: D3, jQuery, jQuery UI, jQuery Tooltip, jQuery Timeago, jQuery Noty, CryptoJS. Some functions provided on StackOverflow were also used. Images were obtained from a number of royalty free sources.

REFERENCES
