Logged Interactive Visualization

Garth Griffin

November 16, 2010

COMP 150-12 Topics in Visual Analytics
Assignment 4

1 Functionality

1.1 File parsing

The loads data by parsing a CSV file. It can take a file as a command-line argument, or the file can be loaded through the GUI. The parser automatically detects the type of data in the CSV file, choosing between floating-point numbers, strings, and categorical data. As before, any value that cannot be parsed is ignored and stored as `null` in the database. The code also recognizes the '?' character as representing a value that is intentionally missing, and stores this as `null` as well.

1.2 Main GUI panel

Once a data file is loaded, the main GUI panel provides the following functionality:

- Display
  - All the columns in the dataset, with names if available, including the type of data in the column (numeric, category, or string).
  - All values in any row (row is chosen with a slider).
- Edit
  - Remove row or column (undoable)
  - Change cell value (undoable), input is constrained to appropriate type (numeric data vs. categorical data)
  - Undo (if there is an action to undo)
  - Redo (if there is an action to redo, choosing between multiple options)
  - Print editing history to the command line (if any edits have been made)

1.3 Pop-out Scatterplot

On the main display there is a button to create a scatterplot. Any number of these can be made. The scatterplot has three dimensions: x, y, and color. Any dimension can be mapped to any column. Edits made to the data will be reflected in all the scatterplots. For numeric data, color is a mapping between red and blue. Mapping color to categorical data will cause the program to use discrete colors for members of each category, rather than a continuum.

1.4 Selection

On any scatterplot, the user can select points by using the mouse to click or click-and-drag. Selecting points on one scatterplot automatically causes the corresponding points on any other scatterplot to become selected as well. By default, selecting new points deselects the points that are already selected. However, this behavior can be controlled with the `Shift` and `Ctrl` keys to change the selection behavior in the following ways:

- `Shift`: Mark newly selected points as selected without clearing the selection (logical OR).
- `Ctrl`: Mark the newly-selected points as unselected without clearing the selection (logical NOT).
- `Shift+Ctrl`: Toggle the selection state of the newly-selected points (logical XOR).

Any selection operation automatically updates all scatterplots.

1.5 Logging

Selection and editing actions (including undo and redo) are logged in a linear format. A panel to print the log is available through the GUI. The log can be printed in three ways:

- Full log: writes every logged interaction in full detail.
- Log as CSV: writes the time, origin window, and type of every logged interaction.
- Selection log as CSV: writes selection events with timestamp, number of points selected, and the visible axes in the window where the selection occurred.

2 Architecture

The program is written in Java. It uses an external CSV parser[1].

2.1 Storing the data

The values are stored in a two-dimensional array based on the Java `ArrayList` encapsulated in a custom class. The custom class stores data in two dimensions as an `ArrayList` of `ArrayLists`. It exposes public methods for getting and setting individual values, methods for getting and setting vectors (where a vector is a row or column), and methods for getting the dimensions of the data.

Changes to the data structure are tracked using a tree where each node represents a reversible change. The tree is initialized with a single root node when the input file is read, and then subsequent actions are

[1] http://opencsv.sourceforge.net/

added to the tree in the form of child nodes. Actions that only display data and do not change it are not added to the tree. Any action in the tree is assumed to be reversible provided the data structure is in the state it was immediately following the action (e.g. adding a row is reversed by deleting the row at the same index as the row was initially added). Similarly, actions can be redone by doing the initial action again, provided the data structure is in the state it was immediately before the undone action was first done.

Only one instance of the entire data structure exists in memory, and it is always in the most recent state. In the tree structure, each node only stores as much information as it needs to undo and redo its particular action. Thus, by traversing the tree structure and applying in order the undo or redo methods of each node in the traversal path to the data structure, the data structure can be brought to any state it was during the course of the editing session.

Since there are different data types in the data structure, there is a wrapper class around the basic 2D array that provides getter and setter methods to translate between the backing array and a string representing the data. For example, the wrapper class translates between the strings representing categories and the numerical values that are actually stored for the categories. This allows the data to be manipulated as numbers, but still easily readable by the human user.

Selection is handled by having an extra hidden column with a boolean flag for each row. Methods to set a particular flag, to set a collection of flags, to get a particular flag, and to get a copy of the hidden column are exposed by the data structure. Methods that add or delete rows also affect the hidden column. By default, new rows are unselected.

2.2 GUI

The program is based on the model-view-controller design pattern, but the view and controller elements are consolidated. Thus, the model (as described in section 2.1) holds all the data, and also maintains the history of interactions. When a change is made to the data, the model fires an event. These events are picked up by all elements of the GUI that display data, and when the events are received the GUI updates its display. The parts of the GUI that accept user interaction (buttons, dropdowns, etc.) make the appropriate method calls to update the model.

2.3 Selection

Selection of rows is handled by events. When a user takes an action that changes which rows are selected, the window in which the action takes place calls the approriate method in the model to update the selection state. This causes an event to be dispatched that signifies a change in the selection has been made. Other windows that display selection listen for this event, and repaint as appropriate.

2.4 Logging

Logging is done by listening for events from the model. When an event fires, the logging module adds a new log entry for that event, recording the current state of the system. Recorded state attributes are specific to the type of event. The logging module exposes public functions to write out the logged data in various formats as described above. These functions are called by the LogDisplayFrame GUI object.

2.5 Packages

The classes are organized into packages, as follows.

- `csv`: contains the code to call the external CSV parser library, and also handles detecting the column types
- `db`: defines 2D data structure and wrapper model, instantiates the CSV parser to read in a file when needed
- `editing`: defines data structure to track changes to the model, instantiated by the data model
- `log`: handles logging actions and writing logs to file
- `main`: contains main method, instantiates data model, GUI and log.
- `selection`: defines selection events and listener interface
- `ui`: all the GUI elements