D Project Description

1 Introduction

"The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it—that's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids. ... [b]ecause now we really do have essentially free and ubiquitous data. So the complimentary scarce factor is the ability to understand that data and extract value from it..."

Hal Varian, Google's Chief Economist [109]

Analysts in all areas of human knowledge, from science and engineering to economics, social science and journalism are drowning in data. New technologies for sensing, simulation, and communication are helping people to both collect and produce data at exponential rates [30, 31, 75]. The increasing scale and accessibility of digital data is an underexploited resource with which we might better understand and improve governance, business, academic research, and even our personal lives. Unfortunately, simply making data available in no way guarantees that it is usable, as even data professionals can experience great difficulty in efficiently managing and making sense of large and diverse data sources. In order to produce real value from data, we must make sense of it. Such *sensemaking*—acquiring insights from data—requires integrating large-scale data storage, access, and analysis systems with contextualized human judgments about the meaning and significance of patterns in the data.

The goal of this research is to enable a broad class of data analysts to more effectively work with data, using novel interactive tools for data transformation and visualization. The aim is to improve the efficiency and scale at which data analysts can work, while simultaneously enabling more people to engage with data by lowering the threshold for non-experts. We propose to target two critical challenges in the data life-cycle: *visualization design* – creating effective visual representations of data to gain insight – and *data wrangling* – assessing data credibility and transforming data into usable forms.

By leveraging human visual processing, visualizations can dramatically aid our understanding of patterns in data [9, 113]. However, creating effective, customized visualizations remains a difficult task, as the best visualizations are often meticulously crafted by skilled human designers [6,21,26,27,54,55,79,100,105–108,112,118]. The Protovis project aims to facilitate *the design and deployment of interactive data visualizations*. We will first develop an expressive declarative language for visualization design. Using the language as a base, we will create interactive visualization design tools that combine direct manipulation editing with automated design routines and evaluation aids based on computational models of human perception.

Another impediment to working with data is transforming acquired data into a usable format suitably free of errors [17, 18, 49, 85]. The Wrangler project aims to facilitate *data wrangling* using *interactive visual interfaces that integrate data transformation and analysis methods*. We will develop new visualization techniques for assessing data quality issues and new interactive approaches for authoring data transformations. Our tool will produce transformation scripts specified in a high-level language that enables transformation reuse, modification, annotation with analysts' rationales, and targeting to multiple runtime environments.

The overarching goal is to advance our collective ability to gain and communicate useful insights from a variety of data sources. Our research will produce *novel visual and interactive techniques* for specifying visualizations and authoring reusable data transformations. It will also contribute new systems that marry automated analyses (e.g., visual perception models, data mining algorithms) with direct-manipulation graphical interfaces. Through both *controlled studies and real-world deployments*, we will evaluate our approaches and distill best practices for facilitating data analysis. These research experiences will also inform our *interdisciplinary educational efforts* teaching technologies and practices for data analysis and visualization.

2 Prior Research Accomplishments

In prior work, I have designed novel visualization and interaction techniques, developed software architectures for visualization, and conducted controlled human-subjects experiments and longitudinal deployments. These experiences provide the background skills necessary to successfully conduct the proposed reesearch.

2.1 Visualization Techniques

Research on visualization techniques applies perceptual principles to create more effective representations of information, often coupled with interaction techniques to enable data exploration. Controlled user studies and visual perception experiments are often applied to discover relevant design principles and evaluate new designs. My prior work has produced a number of novel interactive visualization techniques, including systems for exploring large hierarchies [46], evolving hierarchies over time [10], temporal patterns in genealogical data [64], online social networks [45], and e-mail archives [38]. I have also introduced perceptual optimization techniques for choosing the aspect ratios of charts to enhance trend perception [35] and designed novel animation techniques for improving transitions between visualization views [48]. I evaluate these techniques through controlled experiments demonstrating improved performance on analysis tasks [48, 64], case study deployments with representative end-users [10, 38], or both [45].

2.2 Graphical Perception Studies

I have also initiated a related research agenda renewing empirical work in *graphical perception*: the ability of viewers to decode information in visualizations [2, 14, 15, 32, 67, 70, 73, 82, 97, 98, 101, 104]. My collaborators and I have conducted experiments providing perceptual characterizations and design guidelines for novel time-series visualizations [36], identifying the data densities at which "horizon graphs" (a compact form of time-series presentation) outperform standard line charts. I have also conducted multiple studies of subjects' decoding performance of elementary visual encoding variables [43, 66] such as position, length, angle, and area. One outcome is evidence contradicting a decade-held design assumption [7, 96]: that visualizations involving rectangular area comparisons (e.g., treemaps [96] and cartograms [21]) should try to make all rectangles as "square-like" as possible. These studies have provided new insights for effective visualization design and made methodological contributions by establishing the validity of online crowdsourcing (e.g., via Amazon's Mechanical Turk) for conducting a subset of perceptual studies [43].

2.3 Visualization Frameworks

A consistent challenge in user interface development is finding software tools that enable novel interface designs yet also help minimize development time. To address this challenge in the area of interactive visualization, I developed Prefuse [39, 40, 47], a software architecture for interactive visualization. The goal was to simplify the composition of established methods—such as layout and encoding algorithms, dynamic queries, and zooming—while providing an integrated structure in which to develop novel techniques and domain-specific designs. The resulting architecture consists of a unified data model implemented as a memory-resident column-oriented database, an extensible set of operators for performing visual encodings, and facilities for rendering, animation, and interaction. Prefuse and Flare (a version of Prefuse for the Adobe Flash Player) are available as open-source projects. Over five years, the toolkits have been downloaded over 100,000 times, cited in over 1,000 scientific publications, and used by corporations, academic researchers, students, and hobbyists. Despite this success, the approach to visualization design instantiated in Prefuse is limited by its orientation towards software engineers. As discussed in Section 3.1, the unfamiliar toolkit abstractions and the need for significant programming to achieve novel designs discourage adoption by non-programmers.

2.4 Social Data Analysis

Most visualization research focuses on leveraging an individual's visual perception to facilitate cognition. In practice, however, visual sensemaking is often also a social process [41, 110, 115]. People may disagree on how to interpret data and contribute contextual knowledge that deepens understanding. Furthermore, some data sets are so large that thorough exploration by a single person is unlikely. To explore the potential of incorporating social interaction with visual analysis, I built sense.us, a web application for collaborative exploration of 150 years of United States census data [37]. Sense.us integrates visualizations of demographic data with features for collective analysis. Users can attach commentary and annotations to views, share collections of views, and engage in discussion; novel bookmarking and indexing features facilitate view sharing and reduce cross-talk between related visualization states.

We studied usage of the system through a live deployment and a series of laboratory studies [37], and conducted a content analysis [68] of recorded usage. We found that users often combined their knowledge in cycles of observation and hypothesis to make sense of trends in the data. For example, one observer noted a decline in the number of dentists in the labor force; others then hypothesized possible explanations, including fluoridation of the water supply and increasing stratification between dentists and hygienists. In other cases, users explored topics such as changing

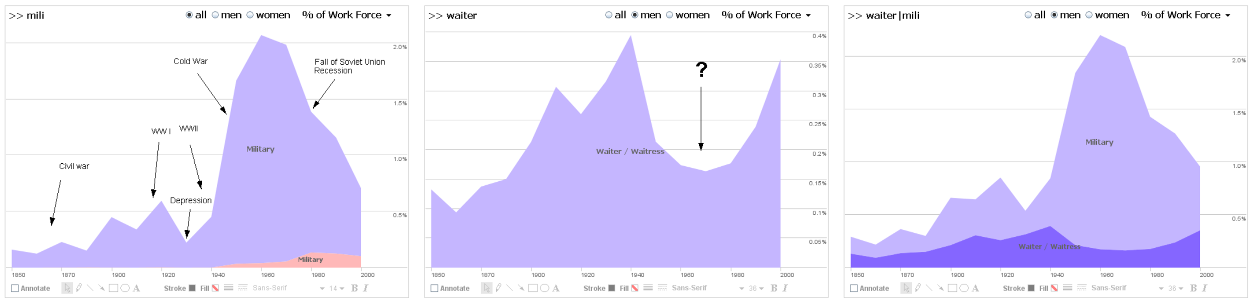


Figure 1: Collaborative sensemaking in sense.us. (a) Users debated the causes of military build-up while (b) another noted a drop and subsequent rise in male waiters in the late 1900's. (c) A different participant connected the two, noting an inverse correlation.

gender roles, the impact of technology on the job market, and correlations among the wax and wane of occupations (Figure 1). We observed that social features helped mobilize users in the process of identifying interesting trends and generating hypotheses, and that exposing social activity regularly catalyzed new explorations by collaborators.

Based on these observations, I designed additional mechanisms to aid collaboration among analysts. *Scented widgets* [120] are user interface controls with embedded visualizations that depict visitation and comment counts for data views reachable from the current application state. I also created *data-aware selection techniques* [42] that tie annotations to underlying data sets by modeling selections as declarative queries. Through controlled experiments we found that these techniques improved data analysis by helping users direct their attention more effectively and annotate data in a more accurate, retargetable manner.

2.5 Relevance of Prior Research

This prior work spanning systems, techniques, and evaluation is relevant for at least two reasons. First, it provides evidence of the skills necessary to conduct the proposed research. Second, the work highlights the importance of the problems we now propose to tackle. Feedback from toolkit users has highlighted obstacles for visualization construction by more general audiences (§3.1), motivating our goal of creating new approaches to visualization design. Our studies of sense.us found that 16% of all commentary concerned data quality [37], as users struggled with interpreting missing data, problematic outliers, evolving schemas, and data integration errors. Our focus on interactive data transformation stems directly from these observed user needs.

3 Protovis: Visualization Design Tools

Researchers and journalists alike have noted that graphical literacy is a critical skill for effectively working with data [17, 27, 109], and thus for making the most of the current "data revolution." Of course, literacy denotes not only the ability to read, but also the ability to write. Accordingly, we need visualization tools that facilitate the design of effective visual representations without over-constraining the design space. Moreover, new visualization design tools should ideally be usable by designers and analysts with varied technical backgrounds, support a variety of deployment contexts (e.g., web and mobile), and scale to large data sets.

In practice, designers choose between many different systems, balancing concerns of *expressiveness* ("Can I build it?"), *efficiency* ("How long will it take?") and *accessibility* ("Do I know how?"). Despite myriad tools for visualizing data, there remains a gap between the notational efficiency of high-level visualization systems and the expressiveness and accessibility of low-level graphical systems. Powerful visualization systems can be intimidating to novices, inflexible in their output, or impose abstractions foreign to visual thinking; this requires designers to translate their intended visual design into toolkit abstractions, often hindering accessibility. On the other hand, graphical systems such as rendering APIs and vector-based drawing programs may be easier to learn, but are tedious for complex work.

My hypothesis is that **we can facilitate the creation of nuanced visualization designs by introducing novel graphical systems tailored to data visualization**. My research will produce and evaluate a visualization design language (named *Protovis*) and related interactive design tools to assess the following hypotheses:

- A declarative language for **the composition of simple, data-representative marks** can achieve a level of *expressiveness* comparable to low-level graphics systems, while improving *efficiency*—the effort required to specify a visualization—and *accessibility*—the effort required to learn and modify the representation.
- A declarative visualization language can provide additional **deployment flexibility and performance improvements**, including the ability to target multiple runtime platforms (e.g., web, desktop, mobile), avoid toolkit-specific data formats, and facilitate unobtrusive performance optimization.
- The use of graphical marks as a language primitive will allow language statements to be authored and revised in **an interactive Visualization Design Environment (VDE)**. A VDE should accelerate design, improve accessibility by minimizing programming, and facilitate publishing and social learning.
- Much how text editing tools provide support in the form of auto-complete or spelling and grammar checkers, a VDE may improve authoring by incorporating **perceptually-motivated automated design routines** and improve evaluation by **visualizing computational models of visual perception**.

3.1 Motivation and Related Work

An array of options exists for creating interactive visualizations. Standard charting programs such as Excel and online tools such as Many-Eyes [111] utilize a *chart typology* [119]: users select from a palette of pre-defined visualizations. While often easy to use, such systems do not permit novel designs or customization. At the other extreme, a programmer may use a graphics API such as OpenGL, Processing or Adobe Flash. With these tools the task of creating a new visualization typically requires a significant software engineering effort in an imperative programming style.

Falling somewhere in-between are visualization frameworks such as the InfoVis Toolkit [25], Improvise [116], prefuse [47], and Flare [39]. The former [25, 116] provide a class hierarchy of visualization widgets; new visualizations are introduced by subclassing an existing component or creating a new one. The latter [39, 47] build visualizations using composable operators for data transformation, layout, and visual encodings. This subdivision enables construction of visualizations in a building-block fashion, similar to data-flow systems in scientific visualization [74]. However, we have found in practice that novel visualizations regularly require programmers to author completely new operators.

An alternative approach is to formulate a *declarative, domain specific language* (DSL) [78] for visualization design. Our hypothesis is that by allowing designers to specify visualizations directly in terms of data-representative graphical marks, we can simplify construction while preserving an expressive design space. By decoupling specification from execution, a declarative approach can also unobtrusively support performance optimization and retargeting across varied runtime platforms.

We take inspiration from existing declarative DSLs such as HTML/CSS [71] and SQL; they are used by millions and insulate users from platform and optimization issues. However, designing a stand-alone language is a difficult task [78] and may complicate integration, as visualizations are often used as components within larger applications. Thus, we believe that the most promising approach is the design of *embedded* DSLs [56]. By implementing a DSL for visualization within a host programming language, designers can use familiar syntactic constructs, leverage the capabilities of the host language, and integrate with other projects. Already, this approach has proven popular in the functional programming community (e.g., [23, 56]) and is used to facilitate massively parallel programming [11, 78].

We are not the first to note the benefits of declarative languages for visualization: researchers have introduced myriad languages for visual analysis. Examples include Wilkinson's Grammar of Graphics [119], Wickham's ggplot2 [117], Slingsby et al.'s HiVE [99] language for hierarchical layouts, and the VizQL formalism of Tableau and Polaris [102]. These languages provide a high level of abstraction and support rapid analysis, but do not provide fine-grained control over graphics and interaction. We target an intermediate level of abstraction that permits a wide array of custom visual designs, embeds within a host language, and yet avoids the tedium of low-level graphics tools. Indeed, we believe Protovis will provide an attractive visualization platform for these high-level visual analysis tools.

In the current work, we will investigate the design and implementation of declarative languages for interactive information visualization, realized as embedded DSLs that utilize functional programming techniques. Next, we will leverage the graphical nature of our language to construct a Visualization Design Environment (VDE): an interactive design tool that minimizes the need for programming. We will subsequently use our VDE as a research platform in which to incorporate perceptually-motivated automated design techniques and explore the use of computational models of visual perception to aid evaluation of visualization designs.

3.2 Initial Progress: The Protovis Language Design

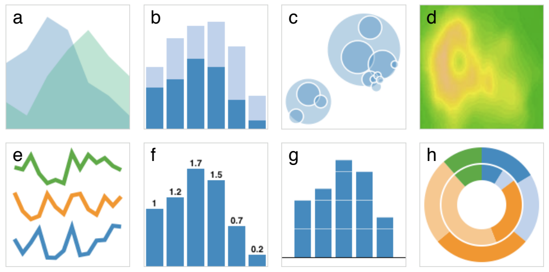


Figure 2: Protovis primitive mark types. (a-h) Area, Bar, Dot, Image; Line, Label, Rule, Wedge.

Protovis takes a graphical approach to data visualization by composing custom views of data with simple graphical primitives like bars and dots. These primitives are called **marks**, and each mark encodes data visually through dynamic **properties** such as color and position. In essence, a mark is simply a collection of bound properties for an associated graphical form. Figure 2 shows these graphical primitives. Most property types are shared across the various mark types to maximize consistency and facilitate lateral movement within the design space.

Marks are associated with data: a mark is generated once per associated datum, mapping the datum to visual properties. Thus, a single mark specification represents a set of visual elements that share the same data and visual encoding. The type of mark defines the names of properties and their meaning. A property may be static, ignoring the associated datum and returning a constant; or, it may be dynamic, derived from the associated datum or index.

Although marks are simple by themselves, one can combine them to make rich, interactive visualizations. To facilitate this, Protovis supports **panels** and **inheritance**. A panel is a container for marks; the contained marks are replicated for each datum on the panel. Protovis uses inheritance to simplify the specification of related marks: a new mark can be derived from an existing mark, inheriting its properties. The new mark can then override properties to specify new behavior, potentially in terms of the old behavior. In this way, the old mark serves as the **prototype** for the new mark. Most marks share the same basic properties for consistency and to facilitate inheritance. Additional details, including language features such as extensibility via "mix-ins", layout algorithms, and interactive behaviors, are provided elsewhere [4].

We have explored multiple implementations of the Protovis model [4, 44]. Our flagship implementation is written in JavaScript and enables web-based visualization; we have also built a prototype implementation in Java to investigate cross-platform deployment (e.g., desktop and mobile) and more sophisticated optimization techniques, including staged compilation, parallel execution and hardware-accelerated rendering. Preliminary results [44] show order-of-magnitude scalability improvements over existing Java visualization libraries.

We have also performed initial evaluations of Protovis [4]. To evaluate expressiveness and efficiency, we built a variety of example applications. Using Protovis, we were able to quickly and concisely specify a diverse set of visualizations. A subset of these examples is shown in Figure 3, including classic visualizations originally drawn by hand juxtaposed with more modern examples, such as interactive stacked graphs and a "mash-up" with online mapping tools. To evaluate accessibility, we analyzed Protovis using the Cognitive Dimensions of Notation framework [34] and solicited feedback from designers, first in a closed beta and subsequently in public releases. The results find that Protovis compares favorably to other popular visualization frameworks such as Flare [39] and Processing [83].

3.3 Research Goals

Our work-to-date on Protovis provides a starting point for research on improved approaches to visualization design. We will investigate the following topics:

Declarative language design for interactive visualization. While we have had success with our initial design and implementation efforts, there remains much to do to further improve the Protovis model. We will extend the model to support animation design, application bookmarking (i.e., exporting visualization state to enable collaborative sharing [37]), inclusion and nesting of more sophisticated layout routines, and a model of composable interactive behaviors for both mouse- and finger-based interfaces. At the systems level, we will continue to further optimize the architecture, including the development of partial evaluation models that avoid unnecessary computation. We will continue to monitor real-world usage and engage with analysts and designers to better support visualization practices.

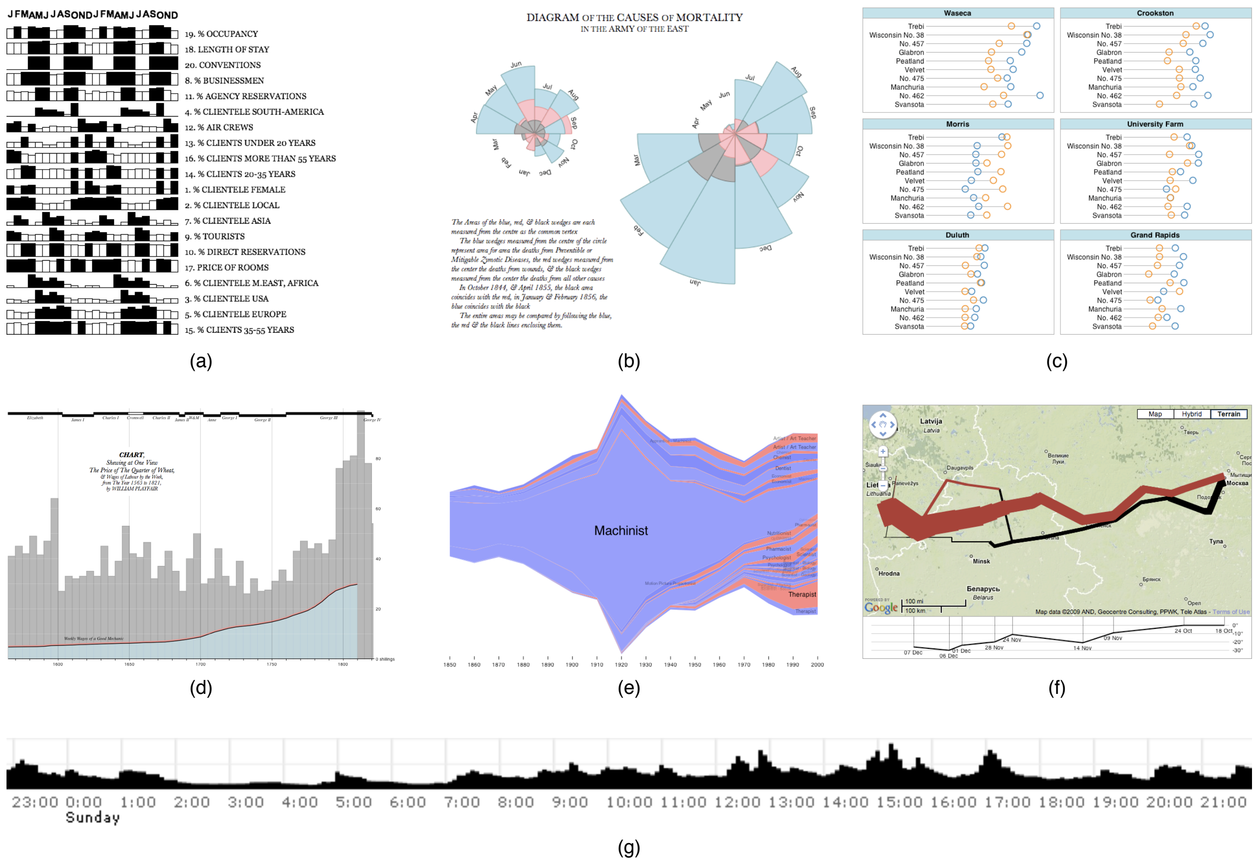


Figure 3: Example Protovis applications. (a) Bertin's analysis of hotel patterns. (b) Nightingale's chart of deaths in the Crimean War. (c) Becker's trellis display of barley yields. (d) Playfair's chart comparing the price of wheat and wages. (e) Heer's Job Voyager. (f) Minard's flow map of Napoleon's march to Moscow, as a Google Maps "mash-up". (g) Live Twitter updates containing the word "oakland". More examples are available at `protovis.org`.

Visualization design environment (VDE). We contend that authoring statements about simple graphical marks provides a more accessible conceptual model than other visualization toolkits: designers can reason directly about the graphic elements with a minimal set of intervening abstractions. While initial work has produced a textual specification language, our conceptual model also naturally lends itself to visual, interactive design tools. We propose to design and evaluate a Visualization Design Environment (VDE) for Protovis, with the twin goals of improving efficiency by accelerating visualization creation and improving accessibility by minimizing programming. Though existing visualization tools such as Tableau/Polaris [102] enable rapid visualization creation, they do so within a closed, formal design space. Analagous to how the Protovis language provides greater expressiveness than closed analytic languages (§3.1), our VDE will enable customized visualization designs and enable designers to invent novel visual representations. We will evaluate our VDE through controlled comparisons with direct programming by skilled Protovis JavaScript users (to assess efficiency) and deployments with non-programmers (to assess accessibility).

To get a sense of what a VDE might be like, imagine an application with a spreadsheet-like interface for manipulating data, coupled with a canvas for direct manipulation of graphics. A designer might select a desired mark type (e.g., a bar) from a palette and draw a bar on the canvas. This gesture would create a new bar mark instance, backed by a data set consisting of a single null value. The drawn extent of the bar defines the default width and height; through interactive manipulation, other default values (position, color, etc) might be set. The user might then drag data from the spreadsheet view onto the bar mark, thereby setting the mark's data property and thus updating the canvas to include as many bars as there are tuples in the data. Using a corresponding property editor, the user could drag data field names onto slots for various visual properties. For example, dragging a numeric data field onto the height property would cause the bars to be sized according to those values. The designer might then change their mind, and use the property editor to change the mark type from a bar to an area: the visualization would then update to show the different representation based on the same data and mappings. In this way, designers can compose marks in a direct manner.

Incorporating automated visualization design. A successful VDE should not only enable users to create data visualizations, it should also help them create more perceptually effective visualizations. Based on principles of graphical perception [15, 36, 43, 113], can visualization design tools help automate parts of the design process? For example, when a user specifies that a data variable should be mapped to a visual variable (e.g., color, shape, size, position) a number of additional design decisions must be made, including scale domains (zero-based or fitted to the data), scale types (linear, log, quantile), and visual ranges (color and shape palettes, element sizing and transparency). Prior work has addressed these issues either through careful choices of default settings [5, 102] or by batch optimization techniques that suggest appropriate visualizations by consulting rank-orderings of the effectiveness of visual variables [76, 77, 89, 90]. Can localized automated design suggestions based on the context of the current visualization state be fluidly incorporated into a designer's interactive session? For example, when initiating a drag of a data field, the system might subtly highlight the drop targets for the visual properties that are predicted to be more effective. When interactively resizing a chart, the system might provide snap-to guides at aspect ratios predicted to be more effective for showing trends in the data [13, 35]. We propose to explore the space of such interactive techniques and assess their effects on visualization design through a series of user studies. This work will be informed by our ongoing research into graphical perception and perceptual design recommendations (§2.2, §6).

Perceptual models to aid evaluation. Our VDE will also provide evaluation tools that help users assess the quality of alternative designs. Word processing tools regularly provide spell checking and grammar suggestions; can a VDE provide analogous assistance based on models of human perception? We propose applying perceptual models to visualization designs and then visualizing the output of these models to inform the design process. For example, vision researchers have developed models for predicting the perception of clutter [87], visual salience and pop-out [60,61,72, 103], perceptual grouping [88,93,95,114], and various forms of color blindness [28]. We will explore different visual representations for communicating the output of such models, such as applying heat maps, color transfer functions, or selective blurring to the current visualization design. As users modify their designs, they should immediately see the updated results of the perceptual model, allowing them to rapidly assess alternatives. We will run studies to observe how designers perceive and incorporate such information and how this impacts visualization design outcomes.

Dissemination and social learning. Finally, we plan to deploy our VDE as a public web application. Users will be able to save and share visualization designs within our service or publish them to the open web, similar in spirit to IBM's Many-Eyes [111]. We hypothesize that this will catalyze a more collaborative exploration of the visualization design space and facilitate the dissemination of novel or customized designs. We will model and analyze the resulting space of visualizations, track temporal trends in usage, and investigate how usage data might be combined with perceptually-driven design suggestions, potentially providing another source of design suggestions with the VDE.

3.4 Research Plan and Feasibility

We have already designed and implemented much of the Protovis declarative language [4, 44]. In the past 9 months the JavaScript implementation has been downloaded over 10,000 times and is now used by a number of companies and universities (§6). We thus believe we have both a strong technical base and supporting community with which to push our research forward. Further research on the underlying software architectures will be on going. We will soon commence creation of the Protovis VDE; once it is in a usable state we will begin alpha, beta, and then public releases of the VDE web application. Throughout this process we will investigate and roll-out features for semi-automated design and perceptual evaluation aids. We will initially conduct controlled studies of these systems and then analyze web-based usage. Should we achieve sufficient web traffic, web analysis will include the results of A/B tests [65].

4 Wrangler: Interactive Data Cleaning and Transformation

I have conducted informal interviews with numerous data analysts and visualization researchers, and nearly all agree on one critical point: **the most painful and time-consuming phase of data analysis is "wrangling" acquired data into a usable form**. First, an analyst must diagnose the data. Is the data responsive to the current analysis questions? What format is it in, and how much effort is required to put it into a format parseable by downstream analysis tools? Are there data quality issues, such as misspellings, missing data, inconsistent values, or unresolved duplicates? Next, the analyst must decide if she wishes to continue working with the data, and if so, she must transform and clean the data into a usable state. Typically, this requires *writing idiosyncratic scripts* in programming languages such as Python and Perl, or engaging in *tedious manual editing* using interactive tools such as Microsoft Excel. It is our hypothesis that this hurdle further discourages large numbers of people from working with data in the first place.

We define such **data wrangling** as *a process of iterative data exploration and transformation that enables analysis*. We say that data is *usable* if it can be parsed and manipulated by computational tools. Data usability is thus determined in conjunction with the tools by which the data will be processed; such tools might include spreadsheets, statistics packages, and visualization tools. Data is *credible* if, according to an analyst's subjective assessment, it is suitably representative of a phenomenon to enable productive analysis. Ultimately, data is *useful* if it is usable, credible, and responsive to one's inquiry. Thus, data wrangling can be characterized as the process of making data useful. Ideally, the outcome of wrangling is not simply data; it is a transcript of transformations and a nuanced understanding of data organization and data quality issues.

Unfortunately, dirty and ill-formatted data constitute an "elephant in the room" of visualization research: the majority of the visualization research literature assumes that input data arrive pristine and so too often turns a blind eye to data formatting and quality concerns. This disconnect suggests a research opportunity, as the need for data wrangling is a common impediment to data analysis which visualization and interaction techniques could do much to alleviate. Outside of visualization research, the database community has developed numerous techniques for cleaning and integrating data. Most of this research focuses on specific data quality problems, such as resolving entities for de-duplication [3, 24, 33, 92]. Some interactive visual tools have been introduced for tasks such as schema matching [86], entity resolution [62], and data cleaning [57, 85]. However, most systems for working with data are non-interactive and inaccessible to a general audience, while those that are interactive often make only limited use of visualization and direct manipulation techniques. Rephrased in HCI terms, these tools often suffer from large gulfs of execution and evaluation [81]: it can be difficult both to apply the tools and assess their results.

My hypothesis is that **we can advance the state-of-the-art by integrating database techniques with visual interfaces for data diagnostics and transformation**. More specifically, I propose to develop novel interactive frameworks for data wrangling and conduct evaluative studies to assess the following high-level hypotheses:

- Data triage and transformation should be **integrated, iterative, and interactive**. Integrating data transformation and analysis routines with appropriate visualization and interaction techniques can make data wrangling more effective, efficient, and accessible.
- Visual representations of data can **improve analysts' identification of data quality issues** and can serve as an **input device for data transformations**. New visualization and interaction techniques might improve analysts' ability to diagnose and subsequently transform data.
- The output of data wrangling should be a **set of data transformations**; transformed data is only a by-product. High-level transformation descriptions will enable repeatability, modification, recording of data provenance, and application within multiple environments (e.g., JavaScript in the web browser, MapReduce, Python scripts).
- Data wrangling effort can be effectively **amortized by social collaboration**. A public corpus of transformations and associated data sets will enable reuse and facilitate retargeting of existing transformations to new data sets. Domain-specific semantic data types enabling verification, reformatting, and transformation (e.g., mapping between zip codes and latitude-longitude pairs) might be authored and shared by analysts.

4.1 Motivation and Related Work

Transforming data sources to a desired data model often requires a series of operations, including splitting data into meaningful records and attributes; reformatting, reshaping, sorting, filtering and normalizing data; deriving new data values through functions and aggregation; and key generation. Often times these operations are performed in an idiosyncratic fashion using spreadsheet tools such as Excel or with scripting languages. Scripting languages and statistical packages are often quite powerful, but usually lack an interactive interface. Dozens of commercial ETL (Extract-Transform-Load) tools are designed to help map data sources to a given schema. While many of these tools provide graphical interfaces, they are often restricted to a few special purpose operations and menu-based interaction, with little to no support for direct manipulation or visualization of data.

The database community has contributed a number of algorithmic techniques and, to a lesser degree, interactive systems for aiding data cleaning and integration. Entity resolution—resolving duplicate records—is a well studied problem within the database and machine learning communities (e.g., [1, 3, 24, 33, 92]). Detecting duplicate records often requires multiple algorithms and domain heuristics. Disambiguation techniques rely on metrics such as edit distance and qgram analysis [33] to identify similar fields across records. Many algorithms exploit contextual information, such as co-occurence of references to records or common linkages within a graph [3]. Learning algorithms use labeled examples of matching and non-matching records to train classifiers [92]. D-Dupe [62] is an interactive tool that highlights potential duplicates and updates its inferences using active learning based on user feedback.

When integrating multiple databases, it is often necessary to map each data set to an agreed-upon schema. Schema matching approaches include both schema-level and instance-level algorithms [84]. Schema-level classifiers utilize name matching (edit distance/qgram [33]), normalization (expanding abbreviations, splitting attribute names), and other heuristics [8]. Instance-level classifiers detect patterns across data values, such as similar formatting to infer column types, relationships between columns, and domain overlap between columns [22]. Some interactive tools also focus on the task of schema matching: IBM's Clio [51] combines user input with Bayes classifiers to semi-automatically map schemas, the Midas [12] project includes interactive tools for exploring potential schema maps, and Robertson et al. [86] developed visualization techniques for matching XML schemas.

Another common problem in data cleaning is detecting erroneous values [18, 49, 53]. Such values might arise due to manual data entry or measurement errors, or be introduced by pre-processing or data integration [49]. Outlier detection can be applied to help flag and correct such problems by comparing values using parametric and non-parametric statistical models of numerical data or frequency counts of categorical values. When the data being cleaned adheres to known semantic data types, domain knowledge can be applied, enabling the use of constraint-based or white list methods to detect problematic values [53]. In our work, we will apply these approaches in conjunction with visualization to facilitate perception of potential problems and interaction techniques to specify corrections.

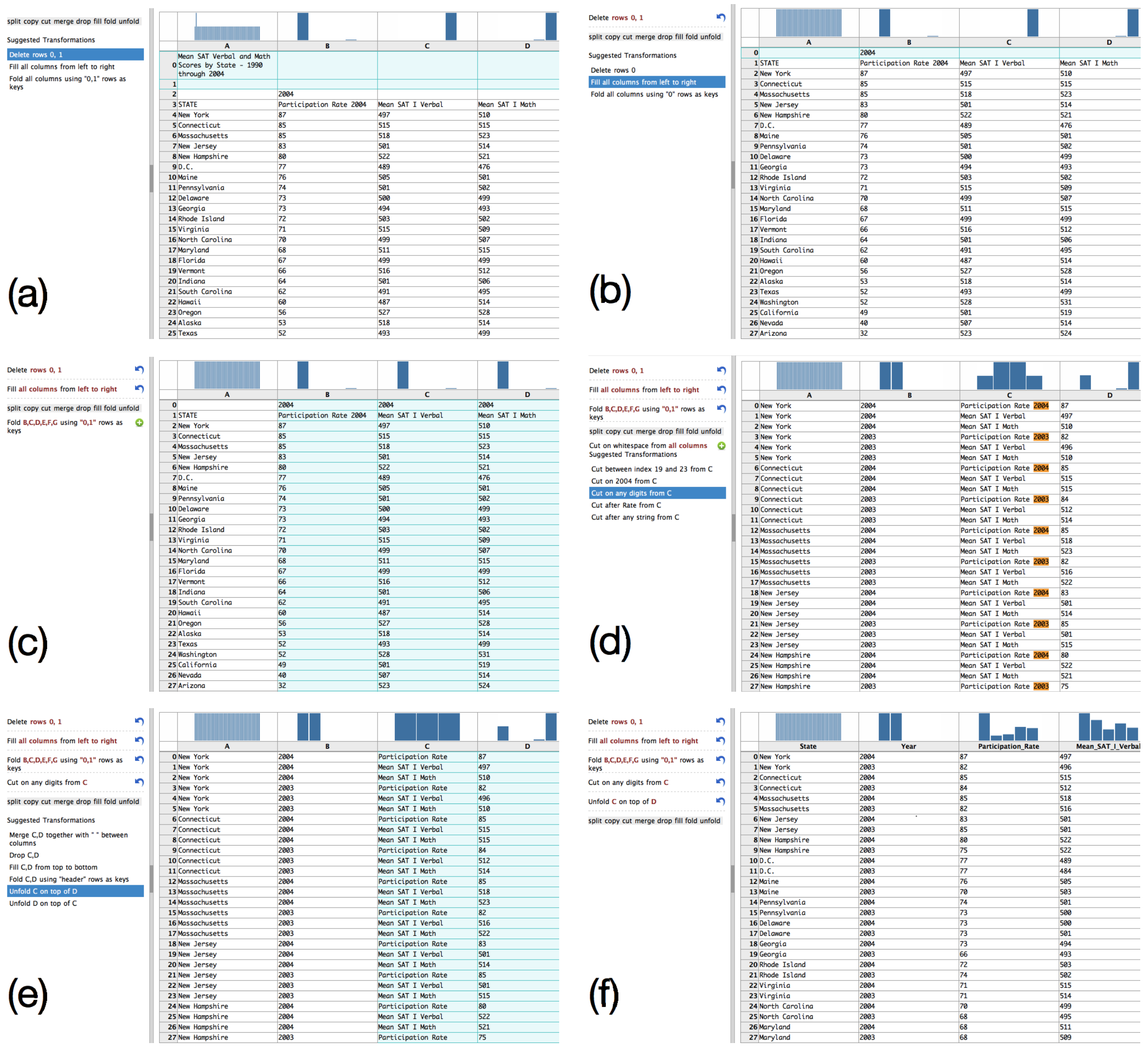
Perhaps most relevant to this proposal is prior work on interactive data cleaning tools. Potter's Wheel [85] provides a transformation language for data formatting and outlier detection. Visualization is limited to a textual spreadsheet and the system does not provide support for multiple tables. GridWorks [57] leverages the Freebase [29] corpus to enable entity resolution and discrepancy detection and provides limited data profiling support through histogram visualizations. However, the system assumes that input data is already in a tabular format. Potluck [58] uses simultaneous text editing [80] to reformat data from multiple sources to a common format. Artemis [52] provides a system for exploring sources of missing data, though its use is restricted to SQL transformations. Our proposed research seeks to more deeply investigate how visualization and interaction techniques can better facilitate these data wrangling tasks.

4.2 Initial Progress: Interactive Reformatting

Consider an example wrangling task, using Standard Aptitude Test (SAT) data from the College Board [16]. The data have been copied out of a PDF file and are not immediately usable by other tools: the data contain extraneous headers and footers, and are formatted as a cross-tabulation by year and measure. Figure 4 illustrates how we can use our current interactive prototype to wrangle the data into a relational format.

First, we select unwanted rows in the data table on the right; the transformation editor on the left suggests possible operations in response (Fig. 4a). We can either hit the Delete key or select the suggested *Delete Rows* transformation. The interface then adds the deletion operation to an interactive transformation history. We then notice that year data in the first row is sparsely populated. We click the row header and execute the suggested *Fill* operation to insert the missing year values (Fig. 4b). We would now like to transform the cross-tab matrix into a relational table; this reshaping involves "folding" the column structure onto itself based on key values. We select all columns except the first, initiate a *Fold* operation, and then additionally select the first two rows to indicate that they contain the key values (Fig. 4c). Executing this transform reshapes the columns into rows containing key values plus a name-value pair.

Next, we clean up the data by removing extraneous year values in column C. Selecting the text "2004" results in multiple interpretations, including matching by index range, by exact content, or any digit string. We choose to *Delete* instances of a digit string (Fig. 4d). We now can "unfold" the data so that each measure is placed in its own column. We select the last two columns and choose the *Unfold* operation (Fig. 4e). Finally, we might also name the columns and specify their data types (e.g., string, date, number) to arrive at the desired, reformatted data (Fig. 4f).



(a)

split copy cut merge drop fill fold unfold

Suggested Transformations

Delete rows 0, 1

Fill all columns from left to right

Fold all columns using "0,1" rows as keys

	A	B	C	D
0	Mean SAT Verbal and Math Scores by State - 1990 through 2004			
1				
2		2004		
3	STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math
4	New York	87	497	510
5	Connecticut	85	515	515
6	Massachusetts	85	518	523
7	New Jersey	83	501	514
8	New Hampshire	80	522	521
9	D.C.	77	489	476
10	Maine	76	505	501
11	Pennsylvania	74	501	502
12	Delaware	73	500	499
13	Georgia	73	494	493
14	Rhode Island	72	503	502
15	Virginia	71	515	509
16	North Carolina	70	499	507
17	Maryland	68	511	515
18	Florida	67	499	499
19	Vermont	66	516	512
20	Indiana	64	501	506
21	South Carolina	62	491	495
22	Hawaii	60	487	514
23	Oregon	56	527	528
24	Alaska	53	518	514
25	Texas	52	493	499

(b)

Delete rows 0, 1

split copy cut merge drop fill fold unfold

Suggested Transformations

Delete rows 0

Fill all columns from left to right

Fold all columns using "0" rows as keys

	A	B	C	D
0		2004		
1	STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math
2	New York	87	497	510
3	Connecticut	85	515	515
4	Massachusetts	85	518	523
5	New Jersey	83	501	514
6	New Hampshire	80	522	521
7	D.C.	77	489	476
8	Maine	76	505	501
9	Pennsylvania	74	501	502
10	Delaware	73	500	499
11	Georgia	73	494	493
12	Rhode Island	72	503	502
13	Virginia	71	515	509
14	North Carolina	70	499	507
15	Maryland	68	511	515
16	Florida	67	499	499
17	Vermont	66	516	512
18	Indiana	64	501	506
19	South Carolina	62	491	495
20	Hawaii	60	487	514
21	Oregon	56	527	528
22	Alaska	53	518	514
23	Texas	52	493	499
24	Washington	52	528	531
25	California	49	501	519
26	Nevada	40	507	514
27	Arizona	32	523	524

(c)

Delete rows 0, 1

Fill all columns from left to right

split copy cut merge drop fill fold unfold

Fold B,C,D,E,F,G using "0,1" rows as keys

	A	B	C	D
0		2004	2004	2004
1	STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math
2	New York	87	497	510
3	Connecticut	85	515	515
4	Massachusetts	85	518	523
5	New Jersey	83	501	514
6	New Hampshire	80	522	521
7	D.C.	77	489	476
8	Maine	76	505	501
9	Pennsylvania	74	501	502
10	Delaware	73	500	499
11	Georgia	73	494	493
12	Rhode Island	72	503	502
13	Virginia	71	515	509
14	North Carolina	70	499	507
15	Maryland	68	511	515
16	Florida	67	499	499
17	Vermont	66	516	512
18	Indiana	64	501	506
19	South Carolina	62	491	495
20	Hawaii	60	487	514
21	Oregon	56	527	528
22	Alaska	53	518	514
23	Texas	52	493	499
24	Washington	52	528	531
25	California	49	501	519
26	Nevada	40	507	514
27	Arizona	32	523	524

(d)

Delete rows 0, 1

Fill all columns from left to right

Fold B,C,D,E,F,G using "0,1" rows as keys

split copy cut merge drop fill fold unfold

Cut on whitespace from all columns
Suggested Transformations

Cut between index 19 and 23 from C

Cut on 2004 from C

Cut on any digits from C

Cut after Rate from C

Cut after any string from C

	A	B	C	D
0	New York	2004	Participation Rate 2004	87
1	New York	2004	Mean SAT I Verbal	497
2	New York	2004	Mean SAT I Math	510
3	New York	2003	Participation Rate 2003	82
4	New York	2003	Mean SAT I Verbal	496
5	New York	2003	Mean SAT I Math	510
6	Connecticut	2004	Participation Rate 2004	85
7	Connecticut	2004	Mean SAT I Verbal	515
8	Connecticut	2004	Mean SAT I Math	515
9	Connecticut	2003	Participation Rate 2003	84
10	Connecticut	2003	Mean SAT I Verbal	512
11	Connecticut	2003	Mean SAT I Math	514
12	Massachusetts	2004	Participation Rate 2004	85
13	Massachusetts	2004	Mean SAT I Verbal	518
14	Massachusetts	2004	Mean SAT I Math	523
15	Massachusetts	2003	Participation Rate 2003	82
16	Massachusetts	2003	Mean SAT I Verbal	516
17	Massachusetts	2003	Mean SAT I Math	522
18	New Jersey	2004	Participation Rate 2004	83
19	New Jersey	2004	Mean SAT I Verbal	501
20	New Jersey	2004	Mean SAT I Math	514
21	New Jersey	2003	Participation Rate 2003	85
22	New Jersey	2003	Mean SAT I Verbal	501
23	New Jersey	2003	Mean SAT I Math	515
24	New Hampshire	2004	Participation Rate 2004	80
25	New Hampshire	2004	Mean SAT I Verbal	522
26	New Hampshire	2004	Mean SAT I Math	521
27	New Hampshire	2003	Participation Rate 2003	75

(e)

Delete rows 0, 1

Fill all columns from left to right

Fold B,C,D,E,F,G using "0,1" rows as keys

Cut on any digits from C

split copy cut merge drop fill fold unfold

Suggested Transformations

Merge C,D together with " " between columns

Drop C,D

Fill C,D from top to bottom

Fold C,D using "header" rows as keys

Unfold C on top of D

Unfold D on top of C

	A	B	C	D
0	New York	2004	Participation Rate	87
1	New York	2004	Mean SAT I Verbal	497
2	New York	2004	Mean SAT I Math	510
3	New York	2003	Participation Rate	82
4	New York	2003	Mean SAT I Verbal	496
5	New York	2003	Mean SAT I Math	510
6	Connecticut	2004	Participation Rate	85
7	Connecticut	2004	Mean SAT I Verbal	515
8	Connecticut	2004	Mean SAT I Math	515
9	Connecticut	2003	Participation Rate	84
10	Connecticut	2003	Mean SAT I Verbal	512
11	Connecticut	2003	Mean SAT I Math	514
12	Massachusetts	2004	Participation Rate	85
13	Massachusetts	2004	Mean SAT I Verbal	518
14	Massachusetts	2004	Mean SAT I Math	523
15	Massachusetts	2003	Participation Rate	82
16	Massachusetts	2003	Mean SAT I Verbal	516
17	Massachusetts	2003	Mean SAT I Math	522
18	New Jersey	2004	Participation Rate	83
19	New Jersey	2004	Mean SAT I Verbal	501
20	New Jersey	2004	Mean SAT I Math	514
21	New Jersey	2003	Participation Rate	85
22	New Jersey	2003	Mean SAT I Verbal	501
23	New Jersey	2003	Mean SAT I Math	515
24	New Hampshire	2004	Participation Rate	80
25	New Hampshire	2004	Mean SAT I Verbal	522
26	New Hampshire	2004	Mean SAT I Math	521
27	New Hampshire	2003	Participation Rate	75

(f)

Delete rows 0, 1

Fill all columns from left to right

Fold B,C,D,E,F,G using "0,1" rows as keys

Cut on any digits from C

Unfold C on top of D

split copy cut merge drop fill fold unfold

	State	Year	Participation_Rate	Mean_SAT_I_Verbal
0	New York	2004	87	497
1	New York	2003	82	496
2	Connecticut	2004	85	515
3	Connecticut	2003	84	512
4	Massachusetts	2004	85	518
5	Massachusetts	2003	82	516
6	New Jersey	2004	83	501
7	New Jersey	2003	85	501
8	New Hampshire	2004	80	522
9	New Hampshire	2003	75	522
10	D.C.	2004	77	489
11	D.C.	2003	77	484
12	Maine	2004	76	505
13	Maine	2003	70	503
14	Pennsylvania	2004	74	501
15	Pennsylvania	2003	73	500
16	Delaware	2004	73	500
17	Delaware	2003	73	501
18	Georgia	2004	73	494
19	Georgia	2003	66	493
20	Rhode Island	2004	72	503
21	Rhode Island	2003	74	502
22	Virginia	2004	71	515
23	Virginia	2003	71	514
24	North Carolina	2004	70	499
25	North Carolina	2003	68	495
26	Maryland	2004	68	511
27	Maryland	2003	68	509

Figure 4: An interactive data wrangling session. The left panel contains a history of transformations and suggested transforms based on the current selection; red text within the transform descriptions indicate parameters that can be clicked and revised. The right panel contains an interactive data table; above each column is a summary visualization showing a histogram of column values. To map SAT data from a cross-tab to a relational format, a user can (a) remove unwanted rows, (b) fill in empty cells, (c) fold columns into rows, (d) remove extraneous text, and (e) unfold rows into per-measure columns. The end result (f) is both a reusable series of transforms and the transformed data itself.

This process results in a wrangling script written in a declarative transformation language (Figure 5). The script provides an auditable description of the transformation and enables later reuse, modification, and verification. Additional work is needed to let analysts annotate these transformations with their rationale. Moreover, the system could perform code-generation to implement transforms in a number of languages; an analyst might wrangle a sample of a large data set, then export transformation code to run on a MapReduce cluster.

```
filter(rows(0,1))
fillHoriz(rows(0))
fold(colRange(1)).keys(0,1)
replace(2).on(/d+/)
unfold(2).on(3)
name(0,1).to('state','year')
```

Figure 5: Data transformation script generated by the wrangling process in Figure 4.

4.3 Research Goals

The previous example serves to illustrate our current research progress and demonstrate how interaction can facilitate data wrangling. Here, I more explicitly outline our research goals.

A declarative data transformation language. Underlying our interactive tools is a declarative language for data wrangling operations. Extending previous systems such as Potter's Wheel [85], the language design includes operators for splitting, merging, extracting, and deleting values; transforming table structures (e.g., fold and unfold); computing aggregates; and using extensible semantic types to transform and verify values. Our language design process is guided by both prior work and empirical data: we are collecting a corpus of data sets from varied sources (e.g., data.gov, NGOs, system log files, and web API results) to derive wrangling requirements. We will initially focus on single tables, then extend the language to support multiple tables, e.g., extracting multiple tables from semi-structured data.

Critically, our language provides a high-level representation of data transformations. As a result, interactive wrangling will produce reusable scripts that can be modified to wrangle a new data set, inspected to communicate data provenance, and annotated to indicate an analyst's rationale. Using a high-level language also allows us to generate code for a variety of platforms; e.g., a wrangler transformation could be translated into MapReduce code or a Python script.

Leveraging semantic data types. Our language will introduce extensible support for semantic data types such as geographic locations, dates, physical units, and currencies. Prior work [57,94] has shown that shallow semantic models can facilitate verification (e.g., flagging inconsistent or outlying values), reformatting, and type transformations (e.g., mapping a state code to a state name, or returning a central lat-lon coordinate). For example, a simple but very common hurdle in data wrangling is mapping from classification codes (for states, counties, products, industries, etc.) to human-readable names. We will explore how type definitions and transformations can be best included in our language design and runtime environments. As discussed later, we will also explore how we can leverage type information to produce more appropriate visualizations.

Interaction techniques for data transformation. We will invent and evaluate interaction techniques with which analysts can fluidly make statements in the wrangler language. We propose to do this by combining *direct manipulation of visualized data*, *automatic inference of relevant operations*, and *iterative refinement of selected transforms*. For example, an analyst may begin by selecting text of interest in a table cell, from which wrangler will generate suggested interpretations of both operations and operands. A text selection might denote matching on indices (positions 10-11), matching on content (the first instance of the text ","), or a generalization thereof (all commas, any punctuation, etc). Users may disambiguate among suggestions by choosing a desired interpretation or providing more example selections (a constrained form of programming by demonstration). Data transformations will be presented to users as natural text descriptions; transformation parameters will be highlighted in the text and can be clicked and edited to refine the transformation. We will initially evaluate our techniques through rigorous application on our corpus of example data sets, followed by controlled studies comparing interface variants to existing solutions such as Excel.

Visualization techniques for data profiling. Concurrent with the design of interaction techniques, we will explore the space of visualizations that best assist data wrangling. For example, in the early stages of wrangling we expect that manipulating a tabular visualization will prove most effective. Once the data format has stabilized, visualization of column values, including highlighting of identified data quality issues, may be preferred. We seek to address a number of research questions. For instance, how do different representations of missing or inconsistent data affect (a) how well analysts can identify and address data quality issues and (b) how analysts reason about trends and patterns in the overall data? Can semantic types be used to generate and parameterize more helpful visualizations, such as maps, timelines, or even tag clouds? What forms of constrained interaction with the visualizations best support data wrangling? For example, numeric data visualizations could support snap-to guides for data quantiles or standard deviations, enabling outlier removal or specification of robust statistics such as midmeans in an interactive, declarative fashion. We will investigate these issues through design explorations and subsequent user studies.

Leveraging social collaboration. Once suitable wrangling tools have been developed and evaluated, we will examine how social interaction and aggregation might further facilitate data wrangling. We will launch a data wrangler web service, first for select groups only and later in a public release. In addition to the wrangler UI, the service will provide access to a corpus of data transformation scripts, helping reduce duplicated effort. By aggregating and analyzing these scripts, we hope to further simplify the wrangling process. For example, we might analyze data set features (e.g., source URLs, character histograms, columns names) to recommend transformations, or apply machine learning techniques to improve automated type induction based on user-provided type assignments. Another opportunity lies in providing mechanisms for user-contributed type definitions: how can we best enable data domain experts to define new types, complete with formatting, verification, and transformation rules? Though type authoring is likely feasible for only a cadre of advanced users, a broad class of analysts might benefit by applying those types to their data.

Scalability. How can data wrangling tools scale to big data? We will implement runtimes for our wrangler language using scalable analytics frameworks such as MapReduce (e.g., Hadoop) and parallel databases (e.g., Greenplum). We will explore client-server designs in which massively parallel backends will execute transforms and then provide data samples and summary statistics back to the wrangler client. We plan to employ sketch-based techniques [19, 59] from the database literature to enable rapid, interactive responses; and will also consider the use of online aggregation [50] to provide incremental, updating results. Scalability challenges will also likely influence the design of our summary visualizations for data profiling. For example, as data set sizes get large, visualizations of individual data points will suffer from overplotting and thus aggregate- and density-based summary visualizations may prove superior.

4.4 Research Plan and Feasibility

We have already taken the first steps in this research, helping establish the feasibility of our approach. We have designed and implemented an initial version of our data wrangler language, and the example scenario in §4.2 uses a functional interface prototype. Anecdotally, we find that we are able to interactively reformat data at a significantly faster rate than using tools such as Excel or scripting languages. Our next step is to further refine our interaction techniques and conduct more formal studies with data analysts. We will then move on to a comprehensive study of visualization techniques for data diagnosis and further develop our framework for semantic types. Throughout this process, we will examine and address scalability issues. Finally, we will deploy web-based services with which to explore the social aspects of data wrangling and collect usage data.

5 Research Timeline and Management

I plan to follow the timeline charted in Figure 6. I will adopt a phased, overlapping approach to enable assessment of individual milestones and refinement of earlier components in response to insights from subsequent sub-projects. The research assumes support for two Ph.D. students: one taking responsibility for the Protovis project and another for the Wrangler project. I plan to advise additional masters and undergraduate students who will also contribute. Both teams will collaborate around web deployments to identify and leverage synergies; for example, users should be able to import cleaned data from Wrangler into the Protovis Visualization Design Environment. Resource needs include development machines for students, server machines for local deployments, and monetary resources for public deployment on cloud computing services such as Amazon S3.

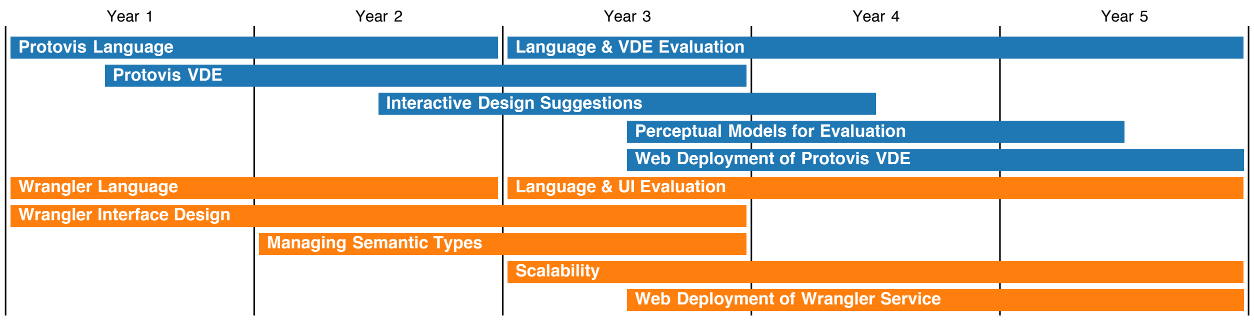


Figure 6: Proposed research timeline. Protovis components are shown in blue, Wrangler components in orange.

6 Outreach and Collaboration

Our visualization and data wrangling tools will be released as freely available, open source software. We hope this will provide significant practical value to data analysts and visualization designers. In prior research, I have found that open source deployment can lead to considerable impact in both academia and industry; my earlier visualization frameworks have been downloaded over 100,000 times and cited in over 1,000 scientific publications according to Google Scholar. Already, the JavaScript implementation of Protovis has been downloaded over 10,000 times and is in production use within Google, IBM, O'Reilly Media, Sandia National Labs, and numerous start-ups; and is in educational use at multiple universities, including Berkeley, Georgia Tech, Harvard, Irvine, Michigan, and UNC. In addition to source code, will we also deploy the Data Wrangler and Protovis VDE interfaces as public web services. These deployments will facilitate broader impacts, provide educational resources, and enable additional experimentation with social computing features.

Moreover, I have a number of active collaborations that will allow us to deploy and evaluate our tools in the context of real-world data life-cycles. We are collaborating with Joseph Hellerstein (a database professor at UC Berkeley) and Greenplum (a parallel database vendor) to address scalability challenges and improve the real-world applicability of our systems. We will work with both in-house analysts and Greenplum customers to better understand analysis issues and solicit feedback on our tools. For example, Greenplum is working with the U.S. Office of Management of the Budget (OMB) to support analysis of reported U.S. government spending data. We will be using this data set—comprised of reported expenditures from diverse government agencies and exhibiting a variety of data quality issues—as one test bed for our data wrangling tools. We are also working with Jeff Pierce and colleagues at IBM Almaden, who, in partnership with local agencies such as the City of San Jose Environmental Services department, is developing data collection tools for citizen scientists. The numerous inconsistencies that can result from public data collection provide another real-world test bed for our data wrangler. Additionally, I am separately conducting research on graphical perception with Maneesh Agrawala at UC Berkeley. The results of our perception experiments will inform automated design decisions within the Protovis VDE.

Within the Stanford campus I have cultivated active ties with Digital Humanities scholars, social scientists, and biologists, among others. Each has data analysis and visualization challenges that the proposed research can help address. For example, in an exploratory engagement we used Protovis to develop visualization tools for field biology data using Prof. Deborah Gordon's 20 years of ant colony observations. Our tools have been used to identify and correct previously unknown data quality errors and facilitate data collection in the field by generating customized maps for research assistants. I will continue to identify and pursue fruitful partnership opportunities.

7 Education Plan

My overarching educational goal is to develop the next generation of data scientists and interaction designers. This aim entails producing leading researchers who will advance computational data analysis practices and competent practitioners skilled at wrangling, analyzing, and communicating data.

7.1 Integrating Learning and Practice

My personal teaching philosophy is that learning is best when paired with doing. My experience has been that project-based courses help students develop practical skills and provide a context in which both theoretical and empirical research can be better understood and applied. As a result, in addition to incorporating research into the curriculum, I strive to **integrate real-world problems and stakeholders with course material**. In my course CS448B *Data Visualization*, I invite guests from both within and outside campus to pitch ideas for potential projects to students. By pairing students with domain experts with real-world data problems, I hope to engage the students in situated learning through *legitimate peripheral participation* [69] and contribute back to the community. In addition to helping students advance their own pre-existing research projects, **course projects have catalyzed a number of new collaborations**:

- Dan Chang, Yuankai Ge and Shiwei Song worked with Digital Humanities scholars in the Stanford Spatial History Lab to visualize the correspondence patterns of Enlightenment-era thinkers. In 2009, they won 1st Prize in the Interactive Maps category at the North American Cartographic Information Society (NACIS).

- John Le, Pao Siangliulue and Jonathan Wang built a visual browser for Professor Fei-Fei Li's Image-Net project [20], an image ontology used for computer vision research.
- Chris Meyer began a continuing collaboration with Professor Gill Bejerano's bioinformatics lab, visualizing computationally-derived mappings between genomic regions and ontology annotations.

Similarly, my course CS294H *Social Software* (co-taught with Google veteran Sep Kamvar) engages both undergraduate and graduate students in a quarter-long process of building a complete social software system, including alpha, beta, and public releases. These systems are often built in collaboration with existing communities or organizations. For example, Coram Bryant, Jacob Klein, and Kelly Nigh worked hand-in-hand with both the Stanford Symbolic Systems program and the Stanford Daily when they created Life Pathways, a site connecting Stanford alumni and students via interviews. At the final project presentations, students received feedback from an invited panel of leading researchers, entrepreneurs, and investors.

As a result of these engagements, my courses have contributed to new **research publications** [4, 43, 63, 64, 91] and **start-up companies and services** (e.g., Alphonso Labs, LineHive.com, RunMonster.com, Unmelt.com).

7.2 Broadening Interdisciplinary Education

When teaching widely-applicable topics such as data analysis and interaction design, **interdisciplinary learning is critical**. For example, visualization integrates knowledge and practices from computer science, psychology, statistics, and graphic design. My courses in data visualization, social software, and human-computer interaction accordingly draw on diverse but complementary domains of knowledge. Though there are many classes that introduce disparate data analysis methods—including courses in visualization, statistics, and data mining—most universities currently lack a course that integrates data analysis needs across the life-cycle of data acquisition, wrangling, visualization, modeling, and dissemination. In collaboration with database professor Joe Hellerstein at UC Berkeley, I am developing **a new course on technologies for supporting the data life-cycle**. We plan to offer the course first at Stanford and then at Berkeley during a subsequent semester.

In conjunction with interdisciplinary material, I believe it is important to **foster an interdisciplinary student body**. First, there is a clear need to teach analysis and interaction design skills to students from disciplines outside computer science, many of whom require (and exhibit) increasing technical skills. Second, I have found that computer science students are enriched by exposure to the challenges and perspectives brought to bear by other disciplines. As a result, I encourage diverse disciplinary backgrounds in my courses; this includes students from psychology, bioinformatics, education, design, music, and journalism. For example, Stanford Knight Journalism Fellows regularly audit my visualization course, and their descriptions of the newsroom environment and its tight deadlines augment students' understanding of the associated design challenges.

Like many faculty, I also believe that broadening education requires **active support for underrepresented groups**. Of my four PhD advisees, one is female and one is African-American; all but one of my masters students are international students, many from developing regions. I also strongly encourage and advise research by Stanford undergraduate and masters students (as our professional masters program does not require research). This support additionally provides valuable advising opportunities for the PhD students.

In addition, I believe more **proactive outreach to other disciplines** is needed to facilitate dissemination of methods and identify new research opportunities. Towards this aim, I help teach a summer course in data analysis methods for social science students. Lastly, our **deployments of research software also constitute an educational opportunity**: through the careful design of examples, online demonstration videos, and social sharing of visualizations and transformation scripts, I hope to use our research to educate and empower interested members of the public.

7.3 Developing Communication and Presentation Skills

In addition to gaining proficiency in a subject, I have found that **fostering strong communication skills** is a critical component of education that is still too-often overlooked, both in the classroom and in advising. As a result, I regularly incorporate student critiques and presentations into my courses. For example, students in CS376 *Research Topics in Human-Computer Interaction* submit critical responses to reading assignments, and rotate as discussants who must present overviews of the papers and lead in-class discussion based on the submitted critiques. Such exercises facilitate critical thinking, better writing, and public presentation skills. Making student submissions visible to the whole course after submission helps foster discussion and promote peer-learning. In CS294H *Social Software*, students must respond to social computing research papers by deriving implications for one or more of the on-going projects in the course. They must also give regular progress presentations. In CS448B *Data Visualization*, students must comment on assigned readings, share critiques of visualization designs, and present and discuss project proposals and progress reports with the class. At the graduate level, developing communication skills also entails delivering research presentations and demonstrations. In addition to regular talks within the department, I also urge students to give guest talks. For example, my group and the visualization and database groups at UC Berkeley conduct exchanges to have students present their research and grow their professional network.

8 Results of Prior NSF Support

I have not previously been supported by NSF. However, two recent proposals have been recommended for funding and are pending final approval. The first proposal is NSF-0964173 *Scalable, Social Data Analysis*, on which I am Co-PI with Maneesh Agrawala (PI) and Joseph Hellerstein (Co-PI) of UC Berkeley. We are investigating collaborative issues in large-scale data analysis practices and supporting integrated annotation and discussion of data sets and derived data products, including interactive visualizations. The research is distinct from, but highly complementary to, the projects discussed in this proposal. The second proposal is NSF-1017745 *Graphical Perception Revisited: Developing and Validating Design Guidelines for Data Visualization*. I am the PI; Maneesh Agrawala is the Co-PI. Our research uses visual perception experiments—conducted both in the laboratory and over the web via crowdsourcing—to characterize the effectiveness of visualization techniques and produce actionable guidelines for visualization design. The outcomes will inform the development of the Protovis VDE described in this proposal.