

SmartCues: A Multitouch Query Approach for Details-on-Demand through Dynamically Computed Overlays

Hariharan Subramonyam, *Student Member, IEEE*, and Eytan Adar

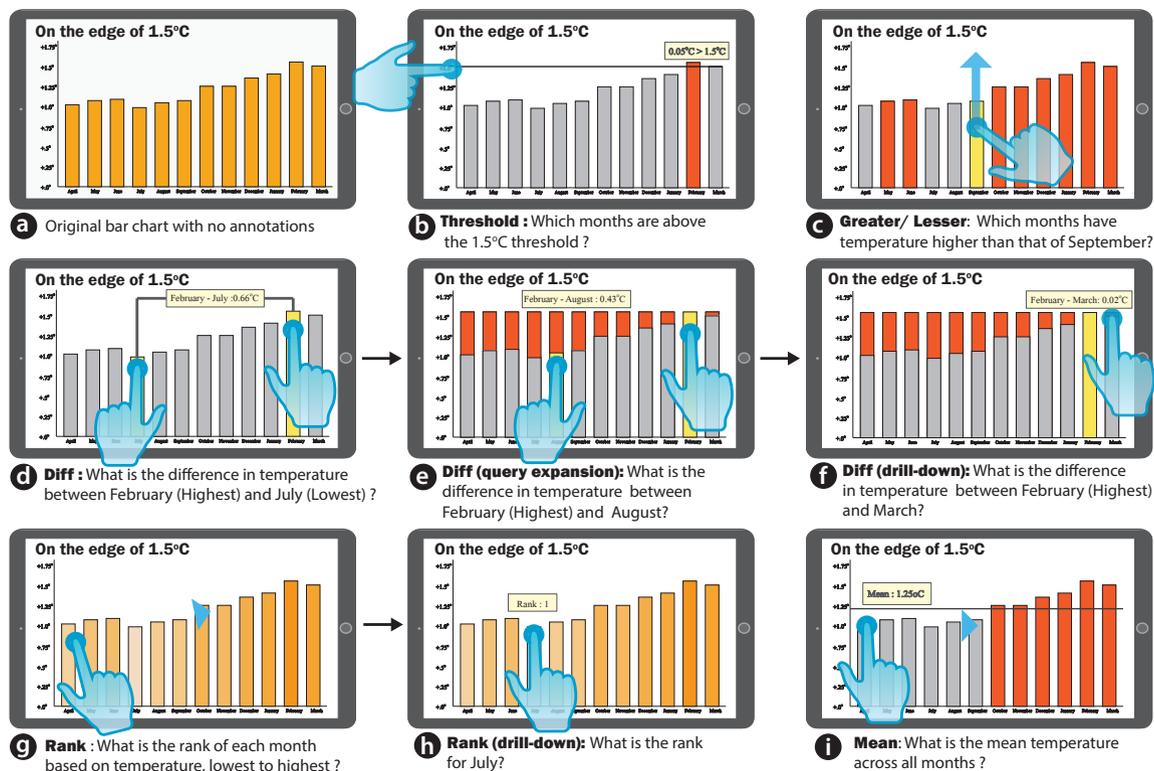


Fig. 1. Query and rendering of SmartCues to boost the effectiveness of a bar chart showing global monthly temperature anomalies for the year 2016 [19]. Each annotation can be called up on demand by performing the corresponding gesture as indicated on the figure. Note that in (b-i) the original bar chart is greyed out for clarity of overlays.

Abstract— *Details-on-demand* is a crucial feature in the visual information-seeking process but is often only implemented in highly constrained settings. The most common solution, hover queries (i.e., tooltips), are fast and expressive but are usually limited to single mark (e.g., a bar in a bar chart). ‘Queries’ to retrieve details for more complex sets of objects (e.g., comparisons between pairs of elements, averages across multiple items, trend lines, etc.) are difficult for end-users to invoke explicitly. Further, the *output* of these queries require complex annotations and overlays which need to be displayed and dismissed on demand to avoid clutter. In this work we introduce SmartCues, a library to support details-on-demand through dynamically computed overlays. For end-users, SmartCues provides multitouch interactions to construct complex queries for a variety of details. For designers, SmartCues offers an interaction library that can be used out-of-the-box, and can be extended for new charts and detail types. We demonstrate how SmartCues can be implemented across a wide array of visualization types and, through a lab study, show that end users can effectively use SmartCues.

Index Terms—Graphical overlays, details-on-demand, graph comprehension

1 INTRODUCTION

The use of details-on-demand (DoD) in interactive visual exploration serves an important role for analysis. Unfortunately, many implemen-

tations of DoD only provide a narrow set of details for a small set of demands (i.e., selection targets). Most conventional DoD implementations use dynamic tooltips that appear when the end-user hovers over a graphical mark. However, there are many situations in which it would be useful to expand both the ‘breadth’ and ‘depth’ of DoDs. For example, when comparing two bars, the end user may ‘demand’ the difference between the two encoded values. On a scatter plot, they may want the distribution between two intervals as detail. Such details are desirable because, as Wilkinson observed, “once coordinates are known and scales calculated, every measurement... is a simple affine transformation for a system that is capable of doing linear algebra in the head,” but that this is, “obviously not human...” [70]. DoDs can pro-

- Hariharan Subramonyam and Eytan Adar are with the School of Information at the University of Michigan, E-mail: {harihars, eadar}@umich.edu

Manuscript received 31 Mar. 2018; accepted 1 Aug. 2018.
Date of publication 16 Aug. 2018; date of current version 21 Oct. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2018.2865231

vide *accurate* access to this information without increasing cognitive load. Critically, DoDs can be presented in a way that does not distract the end-user from analysis tasks by calling attention away from the focus area (e.g., selecting from menus, looking at other panels, etc.). To generalize this idea, we describe SmartCues, an interaction technique and library for dynamically computing and displaying DoDs.

With increasing complexity, DoD displays become difficult to implement and use. ‘Summoning’ a DoD requires that (a) the end-user specifies their selection, that (b) they potentially indicate the type of detail they want (disambiguation), and (c) the detail be displayed appropriately. All three pieces are largely trivial in the case of simple DoD. The end-user hovers over, or clicks on, the shape; the detail information is constrained to a small set of facts (e.g., the value of the bar); and the presentation is a simple tooltip or dynamic annotation. Contrast this to even a slightly more complex task of identifying the mean value for *two* bars in a bar chart (Figure 1d). The viewer must now specify the targets which, if conventionally implemented, would require two clicks. Either the end-user or the system must identify the specific comparison the end-user is interested in or display them all (e.g., do they want the average? the difference? the rank difference?). Finally, the system must determine the best visual representation of this information (is it an annotation connecting the two bars with the value in between? is it a tooltip? is it another visualization? is it an overlaid bar?).

Our solution to this increasingly complex problem is SmartCues. For the end-user, SmartCues supports sophisticated selections and ‘detail queries’ through the (optional) use of multitouch gestures. Detail targets/selection can range from: single elements, such as a bar; global selections, such as all pie segments; and everything in between (e.g., two points or a range in scatter plot). For the developer, SmartCues addresses the complex process of DoD implementation by providing a clear way to build (i) *interactions controls*, (ii) *detail models*, and (iii) *overlay views*. SmartCues is intended to work across a wide array of visualization types, and we have created prototypes for each of the system components. SmartCues can be utilized for both analytic and communicative scenarios. Details are presented as data-aware annotations (e.g., text labels and arrows) [31] and by using ‘un-committed’ retinal channels (e.g., color) or additional marks. Collectively, we refer to such displays as *detail overlays*.

Figure 1 provides an example of SmartCues. The original visualization (a), “On the edge of 1.5°C,” shows the global monthly temperature anomalies in 2016 referenced to the 1881-1910 baseline [19]. Certain perceptual tasks—such as determining which month was highest; determining which month was lower given a pair; or even reading the value for a specific month—are effectively enabled by this display. However, as with all visualizations, this one sacrifices both expressiveness and effectiveness for certain information [47, 73]. Accurately calculating the difference between bars that are very far away is more difficult. SmartCues directly supports this by allowing the end-user to simultaneously tap the two bars (e.g., Figure 1d). This action will produce a graphical overlay with the information. SmartCues can be used to implement many other types of DoD interactions even on this simple visualization. For example, dragging a line from the y-axis (Figure 1b) will indicate a request for a ‘threshold detail’ which displays a color overlay indicating bars exceeding the threshold. Horizontally dragging across multiple bars (Figure 1i) displays an overlay for the mean. A diagonal drag (Figure 1g-h) produces a color overlay for rank. Each detail allows us to ‘recover’ some effectiveness through natural interactions and details display in a salient way. Figure 1c demonstrates a ‘combined’ SmartCues interaction that includes both selection and detail query. By clicking/tapping on a bar (the selection) and gesturing upwards, the end-user is requesting a comparison of all other months in relation to the selection (e.g., higher or lower). While we have created a set of SmartCues for various chart types, the developer/designer can ultimately decide which to enable, disable, modify, or add. Our prototype architecture is intended to support this extensibility.

SmartCues offers additional features to support DoD complexity. To handle many interactions that may become ambiguous in ‘querying’ for a detail, SmartCues both ranks likely detail targets as well as providing interaction methods to resolve ambiguity. SmartCues can also de-clutter

displays that contain too many DoD overlays. Either due to multiple requests or ambiguity, visualizations can become cluttered with too many details, increasing cognitive load and perceptual tasks (e.g., due to label overlap, distracting arrows, etc.). SmartCues works to reduce this clutter by ‘collapsing’ multiple details into alternative views. For example, the end-user may ask for a comparison detail for temperature for February and July (Figure 1d) by touching both bars. If they repeat this for February and August, SmartCues can anticipate that the intent is to compare February to all other months. Instead of showing two comparisons, SmartCues can create an overlay that ‘pre-fetches’ all relevant comparisons and integrates them in a more effective overlay as in Figure 1e. This process is achieved by considering query patterns in predicting future demand.

Our key contribution is introducing a framework by which developers and end-users can have access to complex DoD interactions. We provide a number of extensible implementations of interaction controls (‘WIMP’ and ‘natural’ gestures) and detail models (simple data lookups and statistical functions). By necessity, both interactive controls and overlays are chart dependent. We demonstrate SmartCues on bar charts, line charts, and scatter plots. From these implementations, we identify generalizable techniques and guidelines to adapt SmartCues to new charts. Through SmartCues, we introduce mechanisms for dealing with ambiguity and to ‘compress’ multiple overlays dynamically. We show with a lab-study that SmartCues interactions can be learned and enhance effectiveness for chart comprehension tasks.

2 RELATED WORK

We situate our work in the context of graph comprehension and review current implementations of DoD in facilitating those tasks. We also describe existing annotation systems in the context of DoD.

2.1 Details-on-Demand Interactions

Broadly, *graph comprehension* includes tasks ranging from simple value extraction and pattern detection tasks (i.e., visual queries), to intermediate tasks such as interpolation and finding relationships in data, and advanced extrapolation tasks [6, 11, 25]. Prior work has shown that decoding visual information can be complex and inaccurate [14, 18, 61]. The time taken to interpret a graph is proportional to the number of unique quantitative relations depicted in the graph [14]. For certain tasks, accuracy decreases when the distance between two values being judged increases along the perpendicular axis [18]. Furthermore, while visual structures are better at expressing relational information, tables perform better when tasks involve retrieval of specific values [20]. DoD interactions, in general, address this problem by surfacing data from underlying tables in context of visualizations [59].

Current approaches to DoD can be broadly classified into two types: (1) *selection* based approaches (hover, single- and double-clicks) in which one or more visual objects (marks) are selected to retrieve attribute level details [5, 12, 58, 62], and (2) *zoom* based approaches in which entire visualizations undergo transformation to bring details into view [13, 24, 27]. Segel and Heer [58] analyzed a large number of narrative visualizations in which they find hover-style tooltips to be a common interaction pattern for DoD. Beyond tooltips, details are also presented using sophisticated HTML pop-ups with added interactivity [4, 5], are displayed as table views [62], and even rendered using summary visualizations such as histograms [46, 65]. With SmartCues, we have endeavored to build the same type of support for exploration but have focused on using overlays that do not require secondary windows or visualizations. Rather, SmartCues interactions overlay detail annotations directly onto the original chart.

Zoom-for-detail style interactions take a different approach by loading details dynamically as the end-user ‘magnifies’ the visualization’s drawing surface. Examples include tap-and-hold interactions for scatter plots [13], ontology details in a radial visualization [27], and approaches that re-encode data (e.g., from aggregate to individual marks) during zoom interactions [24]. The Magic-Lens system offers a (zoom) focus+context by using a magnifying-glass ‘lens’ to modify the view under inspection with additional information [64]. The challenge for zoom-based-interactions is that they require reconfiguring the display

and require readers to perform perceptual realignment with the newly adjusted view, which can be costly [40, 44, 50]. In other words, the overall context of the visualization is lost, making it harder to perform subsequent tasks. Additionally, in many domains (e.g., stock price, sensor data), details, as applied to raw data points, are of little interest, and detail level tasks commonly refer to some analytic operation over a subset of raw data [37]. In such cases, details are often presented as an annotation. For these reasons, we focus on ‘in-context’ details.

2.2 Annotations

While not all annotations are details and not all details can be presented as annotations, the overlap is significant. In designing SmartCues interactions and overlays, we leverage existing work on annotation creation and annotation representation. Annotations boost the “natural perceptibility profile” of graphics [1, 22, 45]. Existing work on annotations include both systems [16, 32, 34, 36], and techniques for placement and representation of annotations [1, 2, 17]. Work on placement and aesthetics of annotations (e.g., [1]) have demonstrated that the placement of an annotation is most effective when they coincide with corresponding graphical properties (e.g., graph shape). This work informs our design.

The space of annotations can be broadly classified into three categories: (1) annotations for recording and **communicating** insights (to self or others), (2) annotations that provide **external context**, and (3) annotations that **aid graph comprehension** (the category most related to SmartCues). Orthogonally, we can also classify annotations based on what they ‘bind’ to. The **general** form can include *any* notation on the chart (including ‘scribbles,’ per Wilkinson [70]). The more **specific** variant is bound to data (*Annotation Guides* [70] or *Data-Aware Annotations* [31]). Systems such as Sense.us [32] fall into the *communicative-general* category by offering free-form drawing and textual annotations in collaborative (communicative) settings. Contextifier [34] is an *external-specific* system which automatically generates annotations that connect a temporal visualization (stock data) with a textual story (a news article). In this taxonomy, SmartCues annotations are in the *comprehension-aid-specific* category. SmartCues are connected to data and are intended to boost the effectiveness of visualizations.

Human-driven annotation creation systems such as Click2Annotate [16] and ChartAccent [52] motivate our design of selection and rendering. The designer’s task of *creating* an annotation is similar to the end-users use of DoD (in that similar types of selection and rendering are needed). Click2Annotate takes a semi-automatic approach to instantiate textual “fact” templates (dimension oriented facts, data item-oriented facts, and compound facts) based on end-user selection [16]. While the interaction metaphor is similar to our own, annotations are not overlaid on the graph and are meant for insight externalization and capture. Annotations become “guides” when they are driven by data [70], and the system that is closest to ours in annotation guides is [39]. In Graphical Overlays, Kong and Agrawala present an automated approach to generating annotation guides (reference structures, highlights, numerical data labels, summary statistics, and descriptive text). When ‘correct,’ automated annotations [36, 39] can be informative. However, they may also introduce clutter and noise. DoD approaches, such as ours, are inherently end-user driven in that the annotations are dynamically invoked based on end-user comprehension needs.

2.3 Direct Manipulation Querying Techniques

Direct manipulation (DM) techniques are a natural way of interacting with visual information [60]. They *minimize* the distance between the intent and execution of the intent, particularly in a visualization context [21, 42]. In SmartCues, we take a DM approach to constructing annotation queries. Sadana and Stasko present a set of interactions (selection, zoom, filters) to execute data-centric and view-driven tasks on scatterplots [54]. TouchWave, a system closest to ours, demonstrates multi-touch interactions on stacked graphs to improve legibility and effectiveness of “comparison” tasks between different layers of the stacked graph [8]. In contrast, TableLens fuses symbolic and graphical representations, allowing users to switch between visual patterns and textual details using various “flick” gestures [51].

Other work has focused on extending the capabilities of DM techniques [30, 33], and at developing gestural frameworks for DM querying [49, 71]. Heer et al. present a generalized selection technique that allows users to interactively expand query parameters through query relaxation [30]. In SmartCues, we take a similar “query-by-example” approach to expand the scope of annotation queries. Additionally, for queries with a similar “signature,” we take a mixed-initiative-like approach to allow readers to directly select the desired query operator when confidence in the correct detail is low [33]. While our underlying framework is not restricted to touch gestures, our current implementation uses multi-touch gestures for DM. We are motivated by past work comparing traditional desktop interfaces and FLUID (touch) interfaces, which found that gestural interactions were better suited for problem-solving tasks focused on data [21]. In designing SmartCues, we have incorporated work on eliciting natural gestures [71] and data-aware gestural query specification [49].

3 DESIGN CONSIDERATIONS

SmartCues is composed of three main components that correspond to end-user-facing interactions (interaction controls), internal detail model (functions that calculate what to display), and overlay views (what is displayed). For each, we define a set of guidelines that motivate our design of SmartCues and inform future additions.

Interaction Controls: While many of the interactions described in SmartCues can be implemented through mouse-driven interactions, these are not always the most effective. For the most narrow types of DoD (e.g., a single numerical lookup on a single target), a basic mouse hover may be sufficient to drive a tooltip display. However, with multiple selections and more complex detail options, a standard ‘WIMP’ display may not be effective [42]. Gestural and multi-touch interfaces can support these more sophisticated DoD queries by allowing for multiple simultaneous selections and detail request with few motions. However, gestural interfaces benefit from a correspondence of the gesture to the associated action or selection [42, 71]. That is, gestures should be as **natural (D1)** as possible. Because gestures are potentially ambiguous [57] and can map to multiple possible detail requests, we prefer interactions that **minimize (interaction) ambiguity (D2)**.

Detail Model: The use of gestures, as well as our desire to support a broad array of detail types, complicates the design of our internal detail model. While not all details need be as quickly accessible as mouse-hover based tooltips, ease and rapidity of access is often expected in modern interfaces. A very specific query ‘language’ may result in high precision, but may not be satisfactory in other dimensions such as learnability and cognitive load [29] or access speed. The notion of fast access provides us with the guideline of **reducing time-to-detail (D3)**. An alternative to a formal language, which we adopt, is to model the problem as a search task. A simpler, but more ambiguous, detail ‘language’ can provide the desired simplicity and speed. However, as with interactions, our goal is to **minimize (detail) ambiguity (D4)**.

SmartCues was conceived to support access to a wide array of data-driven details (i.e., those that can be looked up from a table or determined through mathematical, logical or statistical operations on that data). For example, for a bar chart showing sugar quantities for cereal, an end-user may expect that clicking or hovering over the Cheerios bar will display a textual annotation with sugar amount. If the end-user clicks on both Chex and Cheerios simultaneously, they may reasonably expect that the annotation would be the difference in sugar amounts. SmartCues, *does* support a broader definition of details, but these may counter our naturalness guideline (D1). For example, clicking on Cheerios’ sugar bar may display the amount of protein, and clicking on the two bars may display the difference between protein amounts. However, our belief that as these details deviate from what is expressed [47], they become hard for the end-user to anticipate. Put another way, we focus on DoD overlays that **boost effectiveness, not expressiveness (D5)**. Though again, we emphasize that alternative DoD views are possible (e.g., a side panel rather than an overlay), and may be useful if end-users expect such details.

Overlay View: There are a variety of ways details can be produced for display. We restrict ourselves to forms that can be presented in the

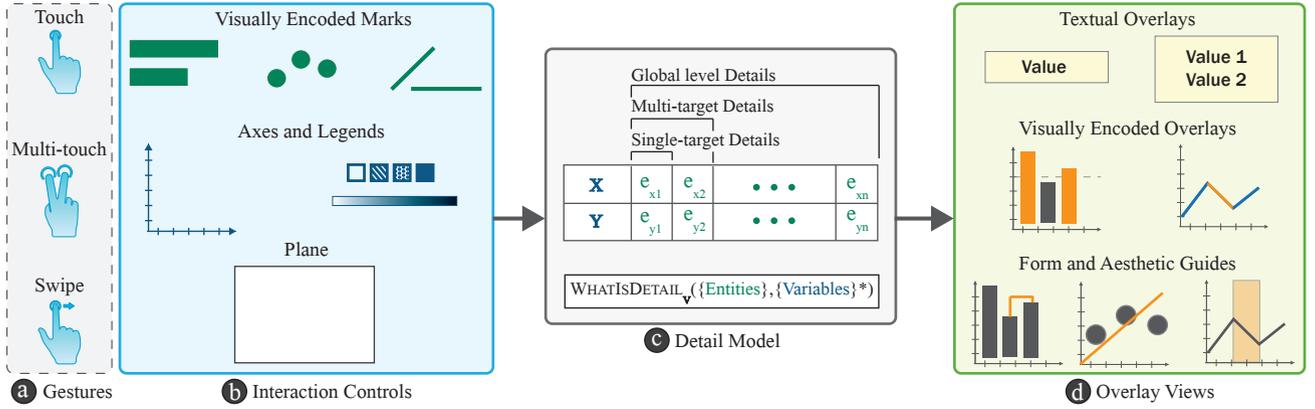


Fig. 2. Overview of SmartCues architecture. (a) Multi-touch gestures allow direct-manipulation interactions with (b) chart controls. (c) Interactions correspond to detail model selection and detail computation. (d) Results (details) are rendered as overlay views.

context of the original visualizations. Consistent with our guideline D3, we prefer for details to be presented rapidly without significant cognitive overhead. Presenting a secondary visualization (e.g., in another window) can increase this cost [35, 69]. Other interactions (e.g., reconfiguration, transformation, filter, etc. [72]) may similarly require a high cognitive overhead to process the new or adjusted view [41, 50]

Rather we adopt the techniques used by others to present details through data-driven annotations [31] and graphical overlays. Both textual and graphical (e.g., trend lines) annotations can be produced by SmartCues. The specific implementation is visualization dependent (e.g., a difference between two pie segments or two bars is represented differently). This approach requires defining a way of presenting details that is uncluttered but with a **clear connection to the selection (D6)** and that **does not collide with retinal variables** (e.g., color) that are already dedicated to encoding values (**D7**). Finally, even though SmartCues ranks details by those most likely to be correct, mistakes need to be resolved. To deal with the inherent ambiguity of the ‘query,’ it is necessary to define mechanisms for displaying different details simultaneously or interactively **resolving (display) ambiguity (D8)**.

4 SMARTCUES ARCHITECTURE

Building on our design guidelines, here we describe each of the three main components of SmartCues (Figure 2).

4.1 SmartCues Detail Model

Before describing the interaction model and graphical overlay, it is worth considering the internal details model of SmartCues. This representation will dictate both how the end-user will query for details and how the details will be displayed.

As shown in Figure 2c, data—bound to a two-dimensional graph—can be characterized by its dependent variable x , the independent variable y , and data entities (data rows) [9, 66]. Detail queries naturally correspond to these three components. Our analysis of existing DoD implementations and data-aware annotations showed that detail types can be mapped to Bertin’s ‘reading levels [9] as well as Ren et al.’s data item, set, and series categories [52]. Elementary level details, such as value look-ups, correspond to a single data entity. Intermediate level details either map to a pair of data entities (e.g., relationship or comparison type details) or may be computed over a series of entities that are subset by independent and dependent variables (e.g., threshold, distribution, etc.). Global level details such as minima, maxima, or mean take *all* of the data entities into consideration. Therefore, at the lowest level, detail queries can be modeled as functions that take two arguments: data entities and variable values. Or more formally: $\text{WhatIsDetail}_v(\{\text{Entities}\},\{\text{Variables}\}^*)$. Because SmartCues details are ‘data-aware,’ the entities argument is required, but variables are optional. The return value of DoD functions are varied. They can range

from ordinal (e.g., what is the rank?) to boolean masks (e.g., which bars are larger than 5?) to vectors (e.g., what is the linear correlation?).

This formalization allows us to express detail queries across all reading levels. For example, $\text{WHATISBARVALUE}(\{bar_v^i\})$ is an elementary question that retrieves the value, v , for entity corresponding to bar i in a bar chart. The question $\text{WHATISDIFFERENCE}(\{bar_v^i, bar_v^j\})$ is an intermediate question that determines the numerical difference between the values encoded by two bars, i and j . The threshold query $\text{WHICHISLARGER}(\{bar_v^0, \dots, bar_v^n\}, k)$ takes an additional parameter k and finds all entities whose value, v , is larger than k . Finally, $\text{WHATISMAX}(\{bar_v^0, \dots, bar_v^n\})$ is a global question that produces the maximum v over all the bars in a bar chart. By specifying detail queries in terms of entities and variables, we also constraint that details be computed over what is already encoded (D5).

A key detail about our specification is that functions may not have a unique signature for any particular graph type. For example, WHATISTHEMEDIAN , WHATISTHEMODE , WHATISTHEMEAN , can all operate on the same selection and will return the same value type. If the request were made through a programming interface, there would be no ambiguity as the function names are all different. Similarly, if the interface had an option for specifying which detail the end-user wanted (e.g., through a menu or drop down box), there would be no ambiguity. However, to satisfy our design guidelines—in particular D3 (reducing time-to-detail)—we provide access to our detail signatures through a ‘fuzzy’ search engine pattern. That is, details can be requested with underspecified ‘queries.’ For example, $\text{WHATIS}^*(bar_v^i)$, will find all detail functions that take one selection as input. This simplifies the task of picking the right function but also introduces ambiguity.

SmartCues attempts to address this in a number of ways (to satisfy D4). By default, SmartCues allows end-users to choose from a list of detail functions that match the query signature (Figure 3g). In addition, we use past selection, and history of detail queries to rank all the matching detail functions. For example, if a query matches prior query signature SmartCues automatically presents that detail based on utility. In other cases, the list is ranked by frequency. Additionally, as discussed in the next section, we can also reduce ambiguity by selecting a good query language.

4.2 SmartCues Interaction Controls

Recall that for our end-users we would like to identify a set of natural interactions (D1) that are simultaneously low-ambiguity (D2) and help maintain fast access to details (D3). Complex menus or additional panels with faceted views can eliminate ambiguity but fail to satisfy our other two objectives.

Rather, we have opted to implement touch-based interactions (Figure 2a) as our ‘query language’ of choice. Our goal is to produce a set of interactions that can be translated into a query applied to the details database (Figure 3). Ideally, such a language would support the selec-

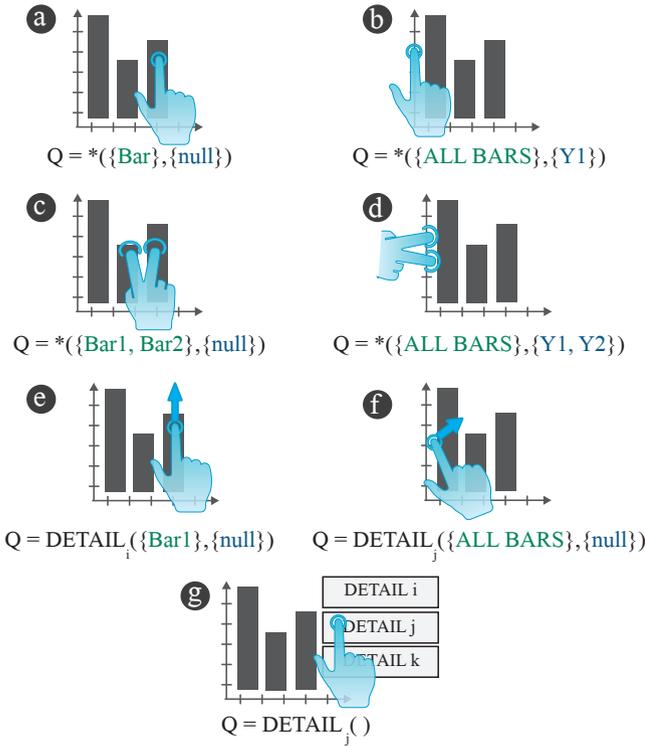


Fig. 3. Gestures and corresponding query expressions for a bar chart.

tion of the data entities, variables/parameters, and the detail function itself. A gesture-based technique can be effective for this purpose as it allows for direct interaction with the chart. The components that define our details model (data entities and variables) directly map to elements of graph composition (Figure 2b). Data entities are visually encoded as ‘marks’ on a two dimensional plane, while independent and dependent variables are represented as axes or legends. Collectively these form the interaction controls (*widgets*) for SmartCues and can be generalized across all chart types.

To guide the design of our gestures we utilized principles of natural scan paths and anchoring (or gaze fixation) from the graph comprehension literature [28, 63, 66]. According to the task anchoring framework [66], graph readers anchor on marks and axes values when extracting different classes of information. For example, to locate a point on a line graph, readers anchor on the x -axis value followed by projection onto the line. Relationship comparison involves pairwise entity anchoring. In SmartCues we align our gestures to analogous visual ‘anchor’ points. The single ‘touch’ (tap) gesture, the most basic of gestures, is used for selecting marks or axes anchor points (Figure 3a-b). This gesture alone covers a variety of DoD tasks across different reading levels. For example, touching on a single bar generates a query $WhatIsDetail(\{bar_i\})$. Similarly tapping on the y -axis value (k) on a bar chart generate a query $WhatIsDetail(\{bar_v^0, \dots, bar_v^n\}, k)$. By design, when the ‘entities’ parameter is not specified by gestures, the query defaults it to all marks on the chart. This both simplifies gesture selections and maintains that queries be data-aware.

To support tasks that involve pairs of marks or range of values, SmartCues also supports simultaneous multi-tap gestures (Figure 3c-d). Tapping on a pair of bars fires a query $WhatIsDetail(\{bar_i, bar_j\})$. Similarly selecting two values on the y -axis generates the query $WhatIsDetail(\{bar_v^0, \dots, bar_v^n\}, y_1, y_2)$. While it is desirable to limit gestures to single and multi-touch, to account for ambiguity, we extend gestures for certain DoD functions as well. Particularly for statistical DoDs such as mean and regression, the query involves all entities and no variables. A simple tap gesture anywhere on the plane would be highly ambiguous. For this reason, we also supports simple ‘swipe’

gestures to reduce ambiguity of detail type (Figure 3e-f). Prior work on gesture elicitation [71] has shown that directional swipe gestures are favored when they correspond to graph semantics (e.g., diagonal downward gesture for downward trends). In SmartCues, directional swipe gestures closely align with visual features of the graph and its corresponding overlays. Tapping on the bar chart and sliding to the right, for example, represents the query for ‘mean.’ Similarly swiping diagonally upwards or downwards translates to ranking queries.

Directional swipe gestures may also be combined with anchor selection when appropriate. For example, tapping on a bar (Figure 3e) and then brushing upwards indicates a request for threshold information relative to the tapped-on-bar. The tap indicates the selection, which would ordinarily be interpreted as a request for details on that bar. However, the upward motion indicates that it is a threshold constraint.

In some situations, a single gesture is unnatural or may be overly complex. For example, we may want to compare the difference between bar_i and all other bars in a bar chart. However, there may not be a single natural gesture for this (i.e., what should the end-user select and how should they move their hand?). A more natural solution is to repeatedly use the $WHATISDIFFERENCE(\{bar_i^i, bar_j^j\})$ gesture (i.e., tapping on bars i , which is fixed, and j which is varied). This achieves the end-users goal but is inefficient. Instead, SmartCues is designed to observe the sequence of queries and infer a higher-level detail request that would ‘encapsulate’ repeated queries. Figure 1d-f illustrates this. After consecutive queries of the same sort, SmartCues will generalize the query from $WHATISDIFFERENCE$ to $WHATISDIFFERENCE TO ALL$. A future extension may be to use generalized selection [30] where repeated tapping or long press expands a selection or detail query.

We have developed a gestural query language consisting of simple tap and swipe gestures (Figure 2a). Because the design of our interactions correspond to anchoring strategies for graph comprehension, the tap gestures carry the same meaning regardless of the chart-type. This makes it easier to implement and use SmartCues on new chart types. However, the directional swipe gestures are chart specific. Notably, the same gesture may mean different things on different charts. For example, a diagonal move from the bottom-left to top-right on a bar chart is a request for rank ordering. The same gesture on a scatter plot is a request for a regression line. While the two share a gesture, the semantics in the context of a chart type is either apparent or can be easily learned. We validated, and modified, our initial gesture set based on feedback from an elicitation protocol (described in Sec. 6.2).

4.3 Overlay Views

A final step in the SmartCues framework is determining the best way to present the details (if the gesture is the query, and details are the documents, then the overlay is the search engine result page). For simple queries (e.g., elementary DoDs) we might imagine placing *textual annotations* near the element the end-user tapped on (similarly to a hover tooltip). As long as the label was near the target or connected by an arrow or some other mark, Gestalt heuristics would lead to a correct association between detail and mark (proximity or connectedness). However, any details that involve more than one target or that have a complex ‘return value’ (e.g., parameters for a regression model) will most likely require different overlay that are specific to chart types. In SmartCues, we have implemented DoD rendering as data-aware overlays. This is consistent with the different grammar-of-graphics style formalism and allow us to layer multiple detail views in the same chart (note that overlays fade).

To inform the design of our overlays, we did a qualitative analysis of existing charts (98 collected from news and visualization sites) and annotation literature [52]. For our analysis, we selected only those visualizations that annotated details about what is already encoded in the visualization. For each annotation, we decomposed (reverse-engineered) it into the data-type of the detail that is annotated, and the function and visualization components that computed (or produced) the annotation. From this, we determined that there are three distinct ‘types’ of detail overlays (Figure 2d). The first, as discussed above, are *textual annotations* and cater to single-valued details. The second type are *visually encoded annotations* and are used to display multi-

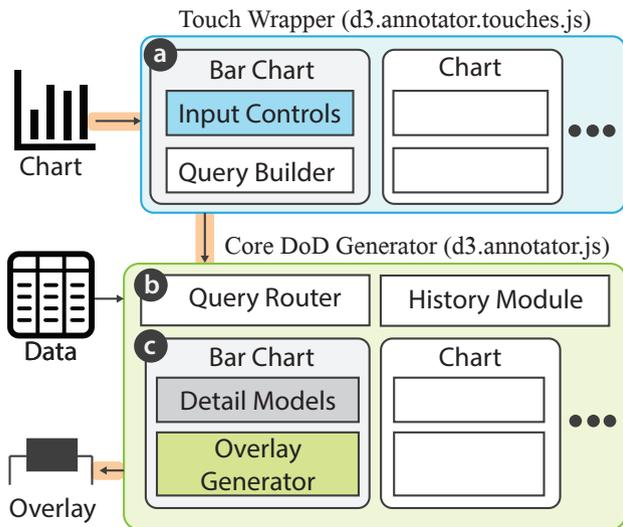


Fig. 4. Overview of the SmartCues library: (a) The touch wrapper binds gestures to charts, and generates detail queries. (b) Core DoD generator module routes the query to the right detail model and (c) charting sub-modules executes the queries and generate DoD overlays.

valued details (e.g., all outliers on a scatter plot, all bars above the mean threshold, etc.). Here, textual annotations alone cannot effectively show the details. To make the information visually perceptible, they are often encoded as modifications to the original mark itself through unused retinal variables (changes in shape, color, etc.). For example, in Figure 1g, textually annotating the rank of each bar is not as effective (for some tasks) as using an overlaid gradient scale (encoding the rank). Labels and gradients can be used, when appropriate, to double-encode the rank information. In this example, the overlay leverages the *unbound* color property of the marks to allow comparison of adjacent marks. It is possible that color or all other visual attributes are used up by the visualization. In such cases, alternatives include encoding the labels, placing additional glyphs, or interactivity. SmartCues overlays are intended to take into account what else is being displayed and how.

The third class of annotations, what we refer to as ‘multi-layered annotations’ refer to overlays that are connected at the data level. For example, a scatter plot that shows an annotation for moving average and also displays the difference between individual points and the average value. This is an example of static multi-layer overlay, but other examples use interactivity to reveal subsequent overlays (e.g., differently colored *outliers* on a scatter plot, whose values are retrieved through tooltips). To fulfill our design guideline D8, we encode complex multi-valued details into higher level annotations, and allow end-users to interactively drill-down to individual details (e.g., Figure 1 e-f).

5 UTILIZING THE SMARTCUES LIBRARY

We implemented SmartCues as a reusable Web-based library. Our implementation offers a collection of detail models, interactions, and overlay views for a set of standard chart types (bar chart, scatter plot, and line chart). More critically, it is possible to adapt and extend this library for other chart types or additional DoDs. Here we briefly describe key implementation details. Complete documentation is available on our demo site at www.smartcues.info.

Figure 4 illustrates the key components of the SmartCues library. The detail models and overlay views are implemented within the core DoD generator library (*d3.annotator.js*) using D3 [10]. The interaction controls are built as a standalone multi-touch library (*d3.annotator.touches.js*). This separation allows for extensibility for other interaction modalities. Execution flows as follows: The touch wrapper (a) captures a valid gesture and invokes the corresponding chart’s query request builder by passing gesture attributes (data marks

and variable values). The resulting request is (b) sent to the routing module of the DoD generator library. Based on the chart type, query specification, and a set of decision rules (multi-layer overlay, history, query generalization support, etc.), the routing module invokes the correct function within the (c) charting sub-module. The results of this function is passed on to the corresponding overlay, which finally renders the computed detail.

5.1 Touch Wrapper

In SmartCues, touch interactions are implemented using the hammer.js [56] and gestrec [43, 67] multi-touch libraries. Given a D3 chart (SVG), this module adds interaction controls to different chart components (marks, axes, plane). When a gesture is invoked, the query builder—a sub-module that handles all gesture events—translates gestures and targets into DoD queries. As shown in Figure 4a, each chart type implements its own interaction controls and query builder sub-modules. The library also offers configuration support in which consumers (end-users or developers) can enable/disable certain DoD queries based on domain context. While a visualization designer can choose to manually add gestures to a visualization, our module reduces the implementation cost from several hundred lines of code to around four lines. As an example, the code to implement all of the touch annotations for a standard (D3) bar chart is:

```
var barAnnotator = d3.annotator.touches('bar')
  .attr("label", <x-axis property>)
  .attr("value", <y-axis property>);
<chart-svg>.call(barAnnotator);
```

5.2 Core DoD Generator

This module handles all of the DoD computation, generation and rendering of overlay views. It consists of three main sub-modules: (1) the query router, (2) a history module, and (3) a collection of chart sub-modules, one per chart type. Each chart sub-module implements a collection of detail models and overlay generation logic. The detail models and overlay views are decoupled so that multiple detail models can invoke the same overlays. When a DoD overlay is requested, the query router performs a predicate match against all detail models for the chart. Additionally, the routing algorithm considers history data and also determines if the query extends an already rendered overlay (drill-down). If a single detail model is matched, the function is directly invoked. If there is a conflict, the router returns a set of potential DoD models which are presented to the end-user through a pop-up widget. The end user can choose which model to apply. Once the details are computed, the overlay module parses the underlying SVG structure to determine the placement and rendering of overlays.

Our library allows for customization of both the detail models and overlays. For detail models, developers can pass in custom functions as predicates, set overlays to manual or auto-clear modes, and configure all of the CSS-style attributes of the overlay including color, thickness, font-size etc. Further, they can extend SmartCues to new chart types by providing detail models and overlay views. Here our core library offers support with placement of overlays and routing gestures.

5.3 Chart-Specific SmartCues

Our library implements annotations and gestures for commonly used chart types including bar charts, scatter plots, and line graphs. These can be used out-of-the-box. Here we describe scenarios to illustrate how SmartCues can support different graph comprehension tasks.

5.3.1 Scenario 1: On the edge of 1.5°C

Figure 1 compares temperature anomalies for the year 2016, drawing attention to the fact that some months are close to or have exceeded the 1.5°C warming threshold—agreed to by COP 21 negotiators in Paris (Climate Central [19]). Starting with an unannotated visualization, the reader first queries for all months that **exceeded the warming threshold** by performing a ‘tap’ gesture at 1.5°label on the y-axis. In response, the core DoD generator renders a SmartCues at that threshold (a horizontal line parallel to the x-axis), and also highlights all bars

NFL field-goal percentage for all kicks vs. average kick distance

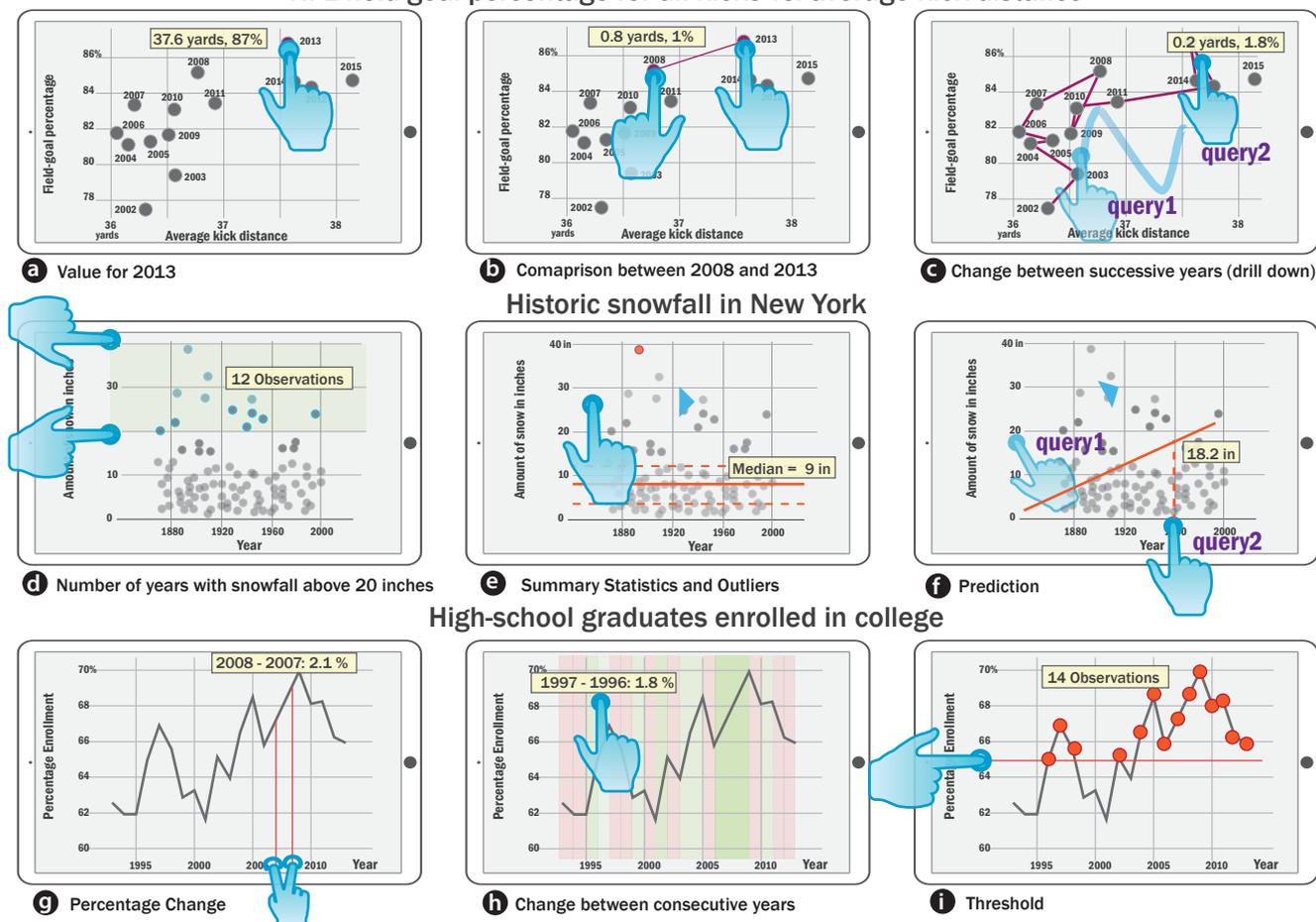


Fig. 5. Example SmartCues showing both interaction controls and overlay views for different chart types. These, and other interactive examples can be accessed from our demo site at www.smartcues.info. Note that the original charts are greyed out for clarity of overlays.

that exceeded that value, in this case February is highlighted (Fig 1b). Next, the reader sees that the temperature increase is not linear with the current ordering of bars. She performs a diagonal swipe-up gesture to get the **ranks of all bars** (Figure 1g). The overlay color encodes each bar on a gradient scale. From here, the reader confirms that July has the lowest rise in temperature by tapping on the bar (Figure 1h). Further, the reader wishes to **compare the highest and lowest months**. She brings up the 'diff' overlay by simultaneously placing two fingers on February and July. This query renders the horse-shoe like overlay connecting the two bars (Figure 1d). Using our query-by-example interaction, she queries for the **difference between February and all other months** (Figure 1e). This tells her that there is only a marginal difference between February and March (Figure 1f). Finally, she queries for the **mean temperature rise** by performing a horizontal swipe gesture across all bars (Figure 1i). The corresponding annotation also shows that six of the twelve months are above the mean value of 1.25. In this manner, the reader is able to explore several facts about this chart, quickly and accurately, using SmartCues.

5.3.2 Scenario 2: NFL Field-goal vs. kick distance

In this visualization, data about field goal percentage is plotted against attempted kick distance for the past fourteen NFL seasons, on a scatter plot (FiveThirtyEight [48]). Here, the reader is interested in comparing kick distances against goal percentage. She first spots that 2013 has the highest field goal percentage. As shown in Figure 5a, she 'taps' on the point to **see the exact kick distance and field goal percentage**. She then uses multi-touch gestures to get the **difference between 2013**

and 2008 which has the second highest goal percentage (Figure 5b). The resulting DoD consists of a line connector between 2008 and 2013 and a label showing the x and y values. Lastly, in this chart there is no apparent spatial ordering of points which makes it difficult for her to **compare changes between consecutive years**. To solve this perceptual issue, the reader performs a zig-zag swipe gesture to bring up a line annotation that connects all years in a sequential order, and then taps on a line segment connecting two years to know the change between the those years (Figure 5c).

5.3.3 Scenario 3: Historic Snowfall in New York

In this third scenario, historic snowfall data for the month of March is plotted on a scatter plot for the city of New York (FiveThirtyEight [23]). Unlike the previous scatter plot in which we examined the relationship between different data points on the plot, in this scenario we look at 'distribution' of points at different levels of snowfall. Starting with an unannotated chart, the reader first wishes to know **how many years saw a snowfall above 20 inches**. She does this by 'touching' on the the y-axis values 20 and 40 (the highest on the scale). This renders a DoD overlay that contains range along with a label showing the exact count (Figure 5d). Next she wishes to retrieve **summary statistics across all observations**. To do this, she performs a horizontal swipe gesture across the chart. The resulting overlay displays median and IQR for the chart along with outliers encoded in a different color (Figure 5e). Lastly, she can examine the relationship between year and amount of snowfall using linear regression. Here she performs a diagonal swipe gesture which overlays the scatter plot with regression line. Further,

she can then predict snowfall for any year by tapping on that year on the x -axis (Figure 5f). This gesture renders a label with the predicted value from the regression equation.

5.3.4 Scenario 4: High school graduates enrolled in college

In this last example, we demonstrate DoDs for a line chart showing percentage of high school graduates enrolled in college across different years (FiveThirtyEight [15]). Here the reader, who is aware of the 2008 recession, is curious to know the **rise in college enrollment compared to previous year** (i.e., 2007). To do this, she simultaneously taps on the two x -axis value. The resulting overlay consists of perpendicular lines from the x -axis which intersect at the corresponding points on the line chart, and also a label showing the actual change between the two years (Figure 5g). Next she wishes to know the same detail for 1996-1997. In response, SmartCues infers query expansion, and generates change overlays across consecutive years (Figure 5h). The overlays are color coded to show rise or fall in enrollment, with darker gradients for steeper changes. The reader ‘taps’ on the overlay at 1996-1997 to see the actual change data. Finally, she wants to know **in which years the enrollment rates were greater than 65 percent?** She taps on the label for 65% along the y -axis which generates a threshold overlay, highlighting years above that value (Figure 5i).

6 EVALUATION

To evaluate SmartCues, we implemented our library on standard chart types including bar chart, scatter plot, and line graph. From an initial pilot we found that because annotations were always ‘right’ (in that they simply reflected the data), correct recall of the gesture was predictive of successfully completing the task. Hence, we focus on two questions: (1) Can end-users easily learn the direct manipulation gestures for different chart types? and (2) Are they able to invoke the right set of annotations to answer chart comprehension tasks?

6.1 Procedure

We conducted our study with 19 participants, a majority of whom were graduate students, and had prior experience working with charts. Twelve participants reported that they engaged with visualizations in the week prior to the study, and six within the past month. Each session lasted 45-60 minutes and participants were paid \$15 for their time. The study was conducted in a lab setting using one of two full-HD multi-touch devices (HP’s Sprout, and Microsoft’s Surface Book).

After completing a pre-test questionnaire, participants received a guided tutorial of the system using a simple, artificial dataset about students. The study coordinator first explained the interaction controls available for each chart type and demonstrated each gesture. Participants were required to practice this gesture on a different machine and successfully complete a set of practice questions. All tasks—training and test—required that the participants interactively query for different annotations similar to the ones illustrated in Figures 1 and 5.

For bar charts, we used a Coffee Sales dataset consisting of sales data for 20 states across the US (also binned into Central, West, East, and South). Example tasks include: “How many states sold more than 40,000 units of coffee?” and “What is the difference in sales between West and all other regions?” For scatter plots, we used an emotion classification dataset (for distribution-related tasks), and a Cricket Chirp-Temperature dataset (for comparison tasks). The speech Emotion Classification consisted of 132 observations plotted by time vs. predicted excitation. The Cricket dataset included 14 observations plotting chirps made at different temperatures. Scatter plot questions included: “At what time interval was the speaker most excited?” and “What is the difference in number of chirps between 57° and 64°?” For line graph tasks, we used another Sales dataset plotting percentage profit across all months (12 data points). Tasks for line graph included: “For the month with the highest drop in profit, by what percentage did the profit decrease from previous month?” The tasks were distributed such that we covered all detail models across chart types, and consisted of ten tasks for bar charts, five for scatter plots, and five for line graphs.

We used more tasks for bar charts to test the large number of DoD features implemented in SmartCues for that chart type.

In addition to the actual responses for each task (collected on paper), we logged the gestures they performed using SmartCues. Finally, all participants filled out a post-test usability questionnaire to report their experience with SmartCues, and provide any open-ended feedback.

6.2 Gesture Elicitation

In addition to evaluating SmartCues broadly, we also tested our gesture choices (to validate our naturalness criteria, D1). The pre-test questionnaire had participants come up with their own gestures for a set of chart comprehension tasks (mean, rank, difference, etc.). Participants were given paper forms in which each task was accompanied by an outline of the corresponding chart type. We first briefed them about different chart components (i.e., marks, axes, and plane). They were then asked to imagine multi-touch interactions for those tasks, and represent them on the chart outline by drawing circles for touch-points, and arrows to indicate motion. As an example, we showed them how they could find the mean on a bar chart by tapping on the left end (circle) of the chart, and by swiping right (right-pointing-arrow from circle). The results from the post-test questionnaire, and our analysis of the pre-test responses, showed that the gestures implemented in SmartCues are identical (16%), or similar (79%), to how participants envisioned them to be. We determined similarity qualitatively based on gesture targets (marks, axes values, and plane) and also type of gesture uses (tap vs. swipe). Gestures were considered different if they corresponded to different interaction controls, and similar if the targets were same as ours but type or direction of gestures were different.

6.3 Results

To measure performance, we analyzed the logs for the set of (time-tamped) gestures performed. We awarded credit for the correct gesture usage, and for any alternate gestures that produced the same result. For example, in Task 11: “*How many observations have a predicted excitation above 70% ?*”, participants opted for repeating the y -interval gesture three times (70–80, 80–90, 90–100) on the scatter plot, instead of directly invoking the y -range query. This may be due to the fact that selecting range was harder (size of touch target) when compared to pointing at intervals. As shown in Fig 6, a majority of participants were able to invoke the correct gestures within a reasonable amount of time. The average time across all tasks and participants was 20.31 seconds ($sd=12.12$ seconds). For Task 9: “*How many products have higher sales when compared to Chamomile?*” several participants failed to invoke the ‘swipe-up’ gesture on the bar of a bar chart. One explanation for this could be that the gesture has the constraint selection (the bar’s value) encoded within the gesture itself, which may make it less intuitive. An alternative may be to first tap on the bar and then swipe. For Task 14, which was a multi-layer annotation interaction, the captured data was noisy because they were triggered on top of overlay.

On a 7-point Likert-type scale (1- strongly disagree to 7- strongly agree), participants rated system along the following attributes: ease-of-use ($\mu=5.57$, $sd=0.76$), learnability ($\mu=5.63$, $sd=1.11$), intuitiveness of task-gesture mapping ($\mu=5.84$, $sd=1.14$), and ease of understanding the annotations ($\mu=6.31$, $sd=0.74$). In providing general feedback about SmartCues, a participant mentioned “[I like]the fact that I could play with the graphs to get information really fast and that I could easily switch contexts for multiple insights...,” and “[I like] the simplicity of system and speed of getting results.” One participant requested support for more complex statistical functions, and a few others expressed the need for better gesture recognition for small touch targets (“*In a scatter plot, I noticed the labels were quite close to each other. So I guess unintentionally, a person may press some other label*”). Lastly we also observed that few participants did not engage both hands while interacting with the system, as one participant expressed “*its hard to hold onto 2 distant bars to get the difference between them.*” We plan to incorporate this feedback in our future iterations through autocomplete-like *cues* for gestures, and also to look at improving multi-touch affordances of different visualizations.

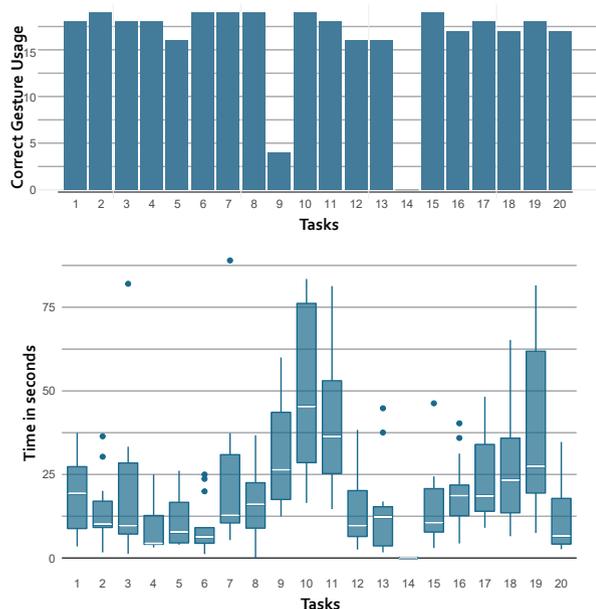


Fig. 6. Results from log data: (Top) Total number of participants who used the correct gestures for each task [$n = 19$]. (Bottom): Time taken for each task across all participants.

7 DISCUSSION

7.1 Assumptions and Limitations

Our implementation of SmartCues formulates DoD as a search process. As with any search logic, the more ‘complete’ the query, the more precise, and unambiguous the response. In our implementation we favor simple touch and swipe gestures that align with visual anchoring processes that are natural to graph comprehension (D1). We trade-off ambiguity (D3) and time-to-detail (D3) for simplicity. While a more complex gestural language may reduce ambiguity, it will likely increase learning cost and may not generalize as well across chart types. We found that ambiguity is low for single and multi-target details as the visual encodings afford specific comparisons and selection targets. However, global-level queries, which do not target specific on-screen marks can lead to ambiguity. Our architecture supports extensions including probabilistic approaches to ambiguity resolutions [57], or alternate query modalities such as natural language (e.g. [26]).

As with any gesture based system, learnability and discoverability pose limitations to SmartCues query language. While the results from our study showed that readers were able to invoke the right gestures post-training, in-the-wild deployment will require additional scaffolding. Specifically, learnability requires that (1) readers develop cognitive mapping between gestures and actions, and (2) are able to perform the gestures with ease [7]. Our gestures are limited to simple touch and swipe, but developing gesture-query mapping can be a challenge. Our design of overlays which closely correspond to gesture targets (D6) could help in establishing such mappings. A possible solution is to use interactive hints (e.g., [38]) and display available gesture icons similar to [3] in order to improve discoverability. Lastly, our current implementation focuses on overlays that yield an effectiveness boost (D5). We do not directly consider how our design plays with visual transformations or other communication-type annotations. However, because our annotations are data-driven, we believe the library can be extended to handle visual transformations and collision with other annotation types. We plan to explore these options in future iterations. By making our library available, we hope to gain additional insights on real-world deployment and use.

7.2 Guidelines for Extending SmartCues

In addition to the general design considerations for SmartCues, here we offer guidelines for extension to novel chart types.

7.2.1 Scope of DoD Queries

Not all SmartCues detail types or gestures need to be enabled for any specific chart or system. Visualizations are selected such that they convey a specific set of insights to readers (comparison, distribution, relationship, composition, etc). Broadly, those insights define the scope for SmartCues. From the reader’s perspective, a simplified view of graph comprehension process is that it consists of *insight acquisitions*—the extraction of expressed facts from the visualization. While typically a ‘visual querying’ process, not all insights can be extracted accurately and quickly by the reader (i.e., effectiveness criteria). When selecting detail models, visualization designers should prioritize those details that have low degree of effectiveness. For example, a ‘rank’ query in an already sorted bar chart may not boost effectiveness. Further, use of SmartCues can lower transformation cost from sort and filter type transformations. Designers should prioritize details that reduce transformation cost. For example, using distribution details overlaid on scatter plots may avoid chart transformation to histograms.

7.2.2 Specification of Detail Models

When defining parameters for detail models, only those parameters that need to be specified by the reader should be included. To allow for simple gestures, other parameters should be assigned a default value. This minimizes time-to-detail without requiring complex queries. For global level queries, detail specifications should be aligned such that the gestures closely comply with all conflicting details (e.g., horizontal swipe gesture for mean, median, and mode). For domain specific visualizations, details can include specific computation that are also visually calculable. For example, a scatter plot of height vs. weight can return the BMI in addition to the specific height, weight values. When detail models align with tasks that a visualization affords, it becomes more intuitive for end users to query.

7.2.3 Integration with other gesture operations

One application for SmartCues is to promote reader engagement with otherwise static visualizations through ‘active reading’ [68]. However, visualization systems commonly require added interactivity such as filtering, brushing and linking, and other view transformation. When such interactions are also implemented using touch gestures (e.g., [53, 55]) they may collide with SmartCues’s query language. One recommendation is to implement SmartCues interactions in a special mode that can be controlled by the reader. This is desirable as SmartCues interactions and overlays are designed to avoid context switches such as view transformations. A second alternative is to explore analogous gestures such as long press instead of tap, pan-start and end instead of two-finger touch, and other naturalistic behavior such as ‘clutching [55]’ the corner of the tablet device to distinguish other gestures from SmartCues queries. A third option, is to further integrate the existing gestures into SmartCues by using our widget menu to handle conflicts.

8 CONCLUSION

Visualizations cannot support all information tasks equally well, partly due to the perceptual and cognitive limitations of the human reader. SmartCues is our solution to boost the effectiveness of visual inferential tasks by making *explicit* what is ‘hinted’ at in the visual encoding. SmartCues are queried using simple touch gestures that closely align with visual scanning processes, and are presented *in situ* to provide quick access to a variety of insights. We demonstrate the approach through a lab study and illustrate how detail models, interaction, and overlays can be added to other chart types. By supporting both broader and deeper notion details, SmartCues provide a new way to satisfy the final step of the mantra: “overview first, zoom and filter, then details on demand” [59].

ACKNOWLEDGMENTS

This work was partially supported by the NSF under grant IIS-1421438. We thank the anonymous reviewers and our study participants for their time and helpful feedback. We also thank Licia He, Matthew Kay, Sile O’Modhrain, and Arjun Srinivasan for their advice and input.

REFERENCES

- [1] C. Acartürk. Points, lines and arrows in statistical graphs. In *International Conference on Theory and Application of Diagrams*, pp. 95–101. Springer, 2012.
- [2] C. Acarturk, C. Habel, and K. Cagiltay. Multimodal comprehension of graphics with textual annotations: The role of graphical means relating annotations and graph lines. In *International Conference on Theory and Application of Diagrams*, pp. 335–343. Springer, 2008.
- [3] C. Ackad, J. Kay, and M. Tomitsch. Towards learnable gestures for exploring hierarchical information spaces at a large public display. In *CHI14 Workshop on Gesture-based Interaction Design*, vol. 49, p. 57, 2014.
- [4] C. Ahlberg. Spotfire: an information exploration environment. *ACM SIGMOD Record*, 25(4):25–29, 1996.
- [5] C. Ahlberg and E. Wistrand. Ivey: An information visualization and exploration environment. In *Information Visualization, 1995. Proceedings.*, pp. 66–73. IEEE, 1995.
- [6] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pp. 111–117. IEEE, 2005.
- [7] F. Anderson and W. F. Bischof. Learning and performance with gesture guides. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1109–1118. ACM, 2013.
- [8] D. Baur, B. Lee, and S. Carpendale. Touchwave: kinetic multi-touch manipulation for hierarchical stacked graphs. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pp. 255–264. ACM, 2012.
- [9] J. Bertin. *Semiology of graphics: diagrams, networks, maps*. University of Wisconsin press, 1983.
- [10] M. Bostock, V. Ogievetsky, and J. Heer. D³ data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.
- [11] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013.
- [12] D. Brodbeck and L. Girardin. Design study: Using multiple coordinated views to analyze geo-referenced high-dimensional datasets. In *Coordinated and Multiple Views in Exploratory Visualization, 2003. Proceedings. International Conference on*, pp. 104–111. IEEE, 2003.
- [13] T. Buering, J. Gerken, and H. Reiterer. User interaction with scatterplots on small screens—a comparative evaluation of geometric-semantic zoom and fisheye distortion. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):829–836, 2006.
- [14] P. A. Carpenter and P. Shah. A model of the perceptual and conceptual processes in graph comprehension. *Journal of Experimental Psychology: Applied*, 4(2):75, 1998.
- [15] B. Casselman. More high school grads decide college isnt worth it. <http://53eig.ht/1tyXZd1>, 2014.
- [16] Y. Chen, S. Barlowe, and J. Yang. Click2annotate: Automated insight externalization with rich semantics. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pp. 155–162. IEEE, 2010.
- [17] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics (TOG)*, 14(3):203–232, 1995.
- [18] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association*, 79(387):531–554, 1984.
- [19] Climate Central. Flirting with the 1.5° c threshold. <http://www.climatecentral.org/news/world-flirts-with-1.5c-threshold-20260>, 2016.
- [20] R. A. Coll, J. H. Coll, and G. Thakur. Graphs and tables: a four-factor experiment. *Communications of the ACM*, 37(4):76–87, 1994.
- [21] S. M. Drucker, D. Fisher, R. Sadana, J. Herron, et al. Touchviz: a case study comparing two interfaces for data analytics on tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2301–2310. ACM, 2013.
- [22] S. Elzer, S. Carberry, and S. Demir. Communicative signals as the key to automated understanding of simple bar charts. In *International Conference on Theory and Application of Diagrams*, pp. 25–39. Springer, 2006.
- [23] H. Enten. New york and philadelphia are in the middle of historically snowy months. <http://53eig.ht/1Ey6k2L>, 2015.
- [24] F. Fischer, J. Fuchs, and F. Mansmann. Clockmap: Enhancing circular treemaps with temporal glyphs for time-series data. *Proc. EuroVis Short Papers, Eurographics*, pp. 97–101, 2012.
- [25] S. N. Friel, F. R. Curcio, and G. W. Bright. Making sense of graphs: Critical factors influencing comprehension and instructional implications. *Journal for Research in mathematics Education*, pp. 124–158, 2001.
- [26] T. Gao, M. Dontcheva, E. Adar, Z. Liu, and K. G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology, UIST '15*, pp. 489–500. ACM, New York, NY, USA, 2015.
- [27] J. Garcia, R. Theron, and F. Garcia. Semantic zoom: A details on demand visualisation technique for modelling owl ontologies. In *Highlights in Practical Applications of Agents and Multiagent Systems*, pp. 85–92. Springer, 2011.
- [28] J. Goldberg and J. Helfman. Eye tracking for visualization evaluation: Reading values on linear versus radial graphs. *Information visualization*, 10(3):182–195, 2011.
- [29] T. Green and M. Petre. Usability analysis of visual programming environments: A cognitive dimensions framework. *Journal of Visual Languages & Computing*, 7(2):131 – 174, 1996.
- [30] J. Heer, M. Agrawala, and W. Willett. Generalized selection via interactive query relaxation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 959–968. ACM, 2008.
- [31] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Commun. ACM*, 55(4):45–54, Apr. 2012.
- [32] J. Heer, F. B. Viégas, and M. Wattenberg. Voyagers and voyeurs: supporting asynchronous collaborative information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1029–1038. ACM, 2007.
- [33] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 159–166. ACM, 1999.
- [34] J. Hullman, N. Diakopoulos, and E. Adar. Contextifier: Automatic generation of annotated stock visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2707–2716. ACM, 2013.
- [35] J. Hullman, S. Drucker, N. Henry Riche, B. Lee, D. Fisher, and E. Adar. A deeper understanding of sequence in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2406–2415, Dec. 2013.
- [36] E. Kandogan. Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pp. 73–82. IEEE, 2012.
- [37] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pp. 9–16. IEEE, 2006.
- [38] B. Kondo and C. Collins. Dimpvis: Exploring time-varying information visualizations by direct manipulation. *IEEE transactions on visualization and computer graphics*, 20(12):2003–2012, 2014.
- [39] N. Kong and M. Agrawala. Graphical overlays: Using layered elements to aid chart reading. *IEEE transactions on visualization and computer graphics*, 18(12):2631–2638, 2012.
- [40] S. M. Kosslyn. Understanding charts and graphs. *Applied cognitive psychology*, 3(3):185–225, 1989.
- [41] H. Lam. A framework of interaction costs in information visualization. *IEEE transactions on visualization and computer graphics*, 14(6):1149–1156, 2008.
- [42] B. Lee, P. Isenberg, N. H. Riche, and S. Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2689–2698, 2012.
- [43] Y. Li. Protractor: a fast and accurate gesture recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2169–2172. ACM, 2010.
- [44] G. L. Lohse. A cognitive model for understanding graphical perception. *Human-Computer Interaction*, 8(4):353–388, 1993.
- [45] R. Lowe and J.-M. Boucheix. Cueing complex animations: Does direction of attention foster learning processes? *Learning and Instruction*, 21(5):650–663, 2011.
- [46] M. Lu, J. Liang, Y. Zhang, G. Li, S. Chen, Z. Li, and X. Yuan. Interaction+: Interaction enhancement for web-based visualizations. In *Pacific Visualization Symposium (PacificVis), 2017 IEEE*, pp. 61–70. IEEE, 2017.
- [47] J. Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141,

- 1986.
- [48] B. Morris. The haters are losing the war on kickers. <http://53eig.ht/1PbD1rX>, 2016.
- [49] A. Nandi, L. Jiang, and M. Mandel. Gestural query specification. *Proceedings of the VLDB Endowment*, 7(4):289–300, 2013.
- [50] S. Pinker. A theory of graph comprehension. *Artificial intelligence and the future of testing*, pp. 73–126, 1990.
- [51] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 318–322. ACM, 1994.
- [52] D. Ren, M. Brehmer, B. Lee, T. Hiller, and E. K. Choe. Chartaccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 230–239, April 2017.
- [53] J. M. Rzeszutarski and A. Kittur. Kinetica: naturalistic multi-touch data visualization. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 897–906. ACM, 2014.
- [54] R. Sadana and J. Stasko. Designing and implementing an interactive scatterplot visualization for a tablet computer. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pp. 265–272. ACM, 2014.
- [55] R. Sadana and J. Stasko. Expanding selection for information visualization systems on tablet devices. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, pp. 149–158. ACM, 2016.
- [56] A. Schmitz, C. Thoburn, and J. Tangelder. Hammer.js. <http://hammerjs.github.io>.
- [57] J. Schwarz, J. Mankoff, and S. Hudson. Monte carlo methods for managing interactive state, action and feedback under uncertainty. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 235–244. ACM, 2011.
- [58] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE transactions on visualization and computer graphics*, 16(6):1139–1148, 2010.
- [59] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336–343. IEEE, 1996.
- [60] B. Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, pp. 33–39. ACM, 1997.
- [61] D. Simkin and R. Hastie. An information-processing analysis of graph perception. *Journal of the American Statistical Association*, 82(398):454–465, 1987.
- [62] J. Soo Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information visualization*, 4(4):239–256, 2005.
- [63] B. Steichen, G. Carenini, and C. Conati. User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pp. 317–328. ACM, 2013.
- [64] M. C. Stone, K. Fishkin, and E. A. Bier. The movable filter as a user interface tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 306–312. ACM, 1994.
- [65] Tableau. Vizable by tableau. <https://vizable.tableau.com/>.
- [66] J. K. Tan and I. Benbasat. Processing of graphical information: A decomposition taxonomy to match data extraction tasks and graphical representations. *Information Systems Research*, pp. 416–439, 1990.
- [67] UW Interactive Data Lab. Gestrec. <https://github.com/uwdata/gestrec>.
- [68] J. Walny, S. Huron, C. Perin, T. Wun, R. Pusch, and S. Carpendale. Active reading of visualizations. *IEEE transactions on visualization and computer graphics*, 24(1):770–780, 2018.
- [69] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '00*, pp. 110–119. ACM, New York, NY, USA, 2000.
- [70] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.
- [71] W. Willett, Q. Lan, and P. Isenberg. Eliciting multi-touch selection gestures for interactive data graphics. In *Short-Paper Proceedings of the European Conference on Visualization (EuroVis)*. Eurographics, 2014.
- [72] J. S. Yi, Y. ah Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, 13(6):1224–1231, 2007.
- [73] Y. Zhu. Measuring effective data visualization. In *International Symposium on Visual Computing*, pp. 652–661. Springer, 2007.