NOAH: Interactive Spreadsheet Exploration with Dynamic Hierarchical Overviews

Sajjadur Rahman[†], Mangesh Bendre[★], Yuyang Liu[+], Shichu Zhu[‡], Zhaoyuan Su[✓], Karrie Karahalios[+], Aditya Parameswaran[#]
[†]Megagon Labs [★]VISA Research [+]U Illinois (UIUC) [‡]Google LLC [✓]UC Irvine [#]UC Berkeley
sajjadur@megagon.ai, mbendre@visa.com, yuyangl2@illinois.edu, shichuzhu@google.com, nick.su@uci.edu, kkarahal@illinois.edu, adityagp@berkeley.edu

ABSTRACT

Spreadsheets are by far the most popular platform for data exploration on the planet. Recognizing increased demand for larger spreadsheets, spreadsheet systems such as Microsoft Excel and Google Sheets now support millions of rows. However, exploring spreadsheets that are this large via operations such as scrolling or issuing formulae can be overwhelming and error-prone. Users easily lose context as they scroll through the data and suffer from cognitive and mechanical burdens while issuing formulae on data spanning multiple screens. To address these challenges, we introduce *dynamic hierarchical overviews* that are embedded alongside spreadsheets. Users can employ this dynamic overview to explore the data at various granularities, zooming in and out of the spreadsheet. They can issue formulae over subsets of data without performing cumbersome scrolling or range selection, enabling users to gain a high or low-level perspective of the spreadsheet data. An implementation of our dynamic hierarchical overview, NOAH integrated within DataSpread, preserves spreadsheet semantics and look and feel, while introducing such enhancements. Our user study demonstrates that NOAH makes it more intuitive, easier, and faster to navigate spreadsheet data compared to traditional spreadsheets like Microsoft Excel, for a variety of exploration tasks; participants make 2.5× fewer mistakes in NOAH while being twice as fast in completing the tasks.

1 INTRODUCTION

With a user base of more than one-tenth of the world's population, spreadsheets are by far the most popular medium for ad-hoc exploration and analysis of data [61]. Studies show that data analysts prefer to operate on data within spreadsheets while shunning BI tools with more advanced analytical features [19, 77]. Spreadsheets enable users to view, structure, and present data in an intuitive tabular layout, wherein users can map their data and tasks; this tabular layout is essential to their popularity [69].

Exploring spreadsheet data using this tabular layout involves two unit operations, scrolling and steering. Scrolling is the action of moving displayed text or graphics up, down, or across a computer screen, in order to view different parts of the spreadsheet. For example, when analyzing data, users may scroll to compare data across screens. Steering, on the other hand, involves clicking the left mouse button and then dragging the mouse pointer through the spreadsheet to select a specific region. For example, to issue a formula, users may steer to select the subset of the data to be operated on as an argument within the formula. Most frequently used spreadsheet formulae require users to perform steering [15, 55]. Overall, both scrolling and steering are crucial exploration primitives for spreadsheets.

Exploring spreadsheet data has gotten a lot more challenging of late: with increasingly large datasets becoming the norm, spreadsheet systems have now raised their scalability limits. Google Sheets now supports five million cells [37], a 12.5× increase from the previous limit of 400K cells, while Microsoft Excel now supports a million rows [2]. Exploring data via scrolling and issuing formulae on spreadsheets with millions of rows can be daunting for end-users, as evidenced by a recent study [102]. *Imagine scrolling through a million rows, a hundred rows (a screenful) at a time—this would take ten-thousand individual scrolling interactions, rendering spreadsheets entirely unusable as a data exploration tool.* Overall, spreadsheet users find it difficult to analyze, make sense of, or operate on such data [69, 93], due to multiple inter-related reasons:

- *Loss of overview and context.* When exploring spreadsheets, users can easily lose context of where they are and where they should go next. The only navigational context provided by spreadsheets is the built-in scrollbar that indicates the user's current location on the sheet. However, since this overview does not capture the layout and structure of the data, users are forced to mentally assimilate the layout and recall it on-demand, as they navigate a spreadsheet.
- *Visual discontinuities.* Users only see a screenful of data at a time. This limited view of the data introduces visual discontinuities between information being displayed. For example, comparing spatially separated subsets of spreadsheet data requires moving back and forth between multiple screens, which can be overwhelming. As an alternative, users tend to copy subsets of data side by side to reduce this visual discontinuity [69], which is cumbersome and error-prone.
- *Cognitive and mechanical burdens.* The lack of contextual cues leads to severe cognitive and mechanical burdens [28]. Users often end up taking drastic measures to avoid getting lost; for example, some users create personalized overviews extrinsic to the spreadsheet, by sketching maps of spreadsheets on paper [93]. Moreover, issuing a simple spreadsheet formula on data spanning multiple screens may require significant manual effort. Users may either select the data via steering, *i.e.*, dragging the mouse pointer, or enter the data range as the argument to the formula after having memorized the range. Both these approaches often lead to errors due to incorrect range selection [74].

Overall, while exploring spreadsheet datasets that span multiple screens, ***users often lose context, experience visual discontinuities, become overwhelmed, and introduce errors***. Despite spreadsheets being around for nearly five decades, little work has addressed the aforementioned challenges that stem from navigating and making sense of datasets spanning several screens. Existing spreadsheet features such as pivot tables, named ranges, and subtotals, partially alleviate some of these challenges but do not eliminate them. For example, pivot tables generate a static summary in a separate area of the spreadsheet while losing the correspondence between the raw data and the summary; named ranges require users to manually label and associate names with ranges of data, a replacement for sketches made on paper [93]. There has been some work on making spreadsheets more scalable [9, 79], and more responsive during updates [12, 80], as well as more expressive [4, 5, 13, 91], but none of these papers address the fundamental usability challenges underlying spreadsheet data exploration, that occur even on modest-sized datasets of a few million rows that all current spreadsheets support. Moreover, while the database community has been increasingly concerned about usability over the last decade, e.g., [3, 26, 30, 43, 46, 56, 68, 87, 100], none of this work directly applies to the spreadsheet context.

So, how do we support more effective exploration of spreadsheet data? One approach would be to integrate an overview of the overall structure of the data along with the spreadsheet [39], resulting in what is known as an *overview+detail* interface [28], where the spreadsheet is the detailed view. Overview+detail interfaces have been shown to be effective in various domains such as text editors and maps [28], reducing cognitive load for users during navigation by providing them with the big picture first, allowing users to zoom in and zoom out on demand. Such interfaces combine fine-grained information into coarse-grained high level groups, *e.g.*, combining states to form countries in Google maps, helping users quickly assimilate the information space [28].

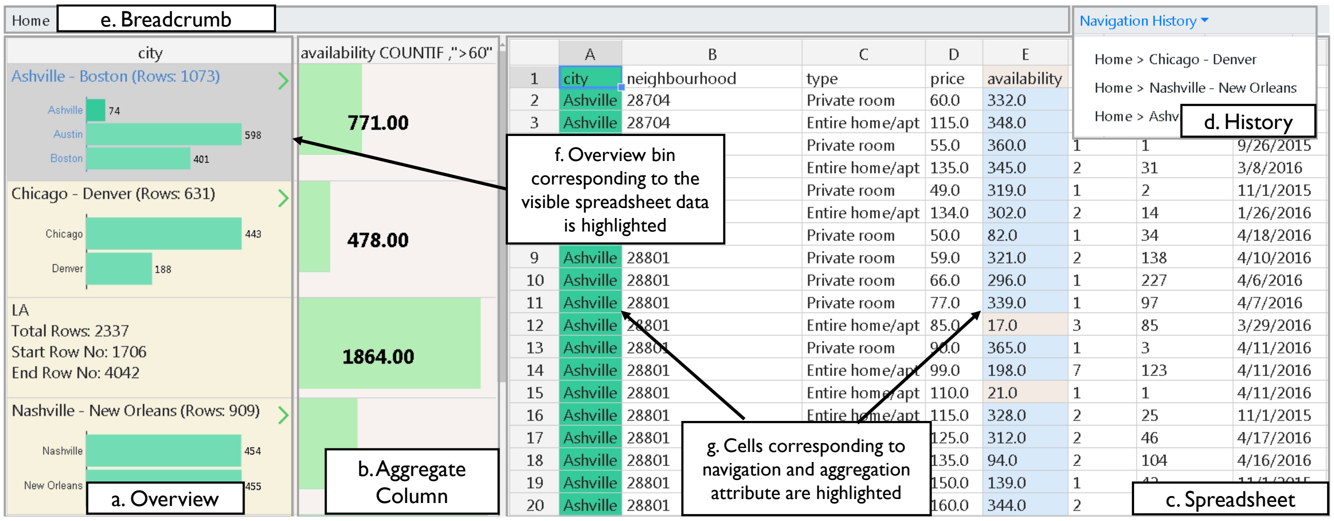
While an overview for spreadsheets, supporting interactions such as zoom in and zoom out, does seem natural, developing it leads to several challenges. The main challenge is that *simply providing a zoomed out view of the spreadsheet as an overview is unreadable and therefore unusable.* Thus, the overview must provide a customizable summary of the data in the spreadsheet, while allowing users to connect the summary to the raw spreadsheet data.

- *Overview design and construction.* Given a large spreadsheet, how do we *design an overview* that captures the overall structure of the data while providing context and reducing visual discontinuities as users navigate the spreadsheet? How do we ensure that this overview *facilitates the search* for individual rows or groups of rows of interest to the user? How do we *construct a coarse-grained representation* of the overview by grouping spreadsheet rows together such that this grouping applies to all data types? How do we *dynamically adapt* the grouping modality so that the overview remains interpretable as users zoom in and out of the overview? If an automatically generated grouping is not semantically meaningful, how do we allow the users to *customize the grouping*?
- *Overview capabilities and integration.* It is essential that the overview we construct is a plug-in that enhances the capabilities of spreadsheets that so many people use and rely on, as opposed to a potentially jarring or confusing replacement for spreadsheets. How can we *seamlessly integrate* our overview with any current spreadsheet tools without impacting existing functionalities or look-and-feel? How do we *facilitate simple interactions* on the overview that help users avoid endless scrolling during spreadsheet exploration and error-prone steering while issuing formulae across multiple screenfuls of data? As user perform various interactions, how do we *ensure consistency* across both views, *i.e.*, the overview and the raw spreadsheet?

Dynamic Hierarchical Overviews. We address the aforementioned challenges using a *dynamic hierarchical overview* which is presented to the user alongside the spreadsheet. This overview is *hierarchical*, in that it allows users to zoom in and zoom out on demand, based on automatically constructed (yet customizable) bins. The overview is *dynamic* in that the overview displays summarized information that is automatically updated as the user performs interactions on the spreadsheet and the overview. Moreover, actions on the overview and the spreadsheet are *coordinated*; users can use whichever one is more convenient to navigate through the data. Finally, users can avoid cumbersome steering operations for formula computation and simply examine a *summarized* view as part of this overview, which are automatically kept up-to-date as the user zooms in and out of the sheet.

NOAH: Our Implementation. We have implemented our dynamic hierarchical overview in NOAH, an in-situ **n**avigation interface for **o**verviewing and **a**nalyzing spreadsheet data **h**olistically. NOAH is constructed as a plugin to an existing spreadsheet tool, DataSpread [10], an open-source scalable web-based spreadsheet. While NOAH's design is not tied to DataSpread, we opted not to use other popular spreadsheet tools like Google Sheets and Microsoft Excel because they are closed-source. NOAH enables new workflows for performing spreadsheet tasks involving scrolling and steering in a rapid and error-free fashion.

Figure 1 shows a snapshot of NOAH. When the user chooses to explore the data by a specific attribute, a hierarchical overview is constructed and displayed within NOAH, next to the raw spreadsheet data (Figure 1a). Users can zoom into or out of the overview to obtain a fine or coarse-grained perspective of the data distribution, similar to drill-down or roll-up operations in data cubes [38]. The distribution at each granularity is captured by automatically constructed histograms, enabling users to assimilate the data via summary statistics. Each bin (group) of the histogram is mapped to a collection of rows in the spreadsheet. Cumbersome scrolling operations are eliminated in favor of a few clicks on the overview interface. Instead of cumbersome and error-prone steering to analyze the data, users can issue formulae directly on the overview with interactions similar to pivot table construction. Users can view results on a separate *aggregate column*—much like aggregation results in cross tabs [38], or aggregates in `GROUP BY` queries—alongside the overview (Figure 1b). Users can issue formulae on different subsets of the data while remaining on the same screen, reducing visual discontinuity. NOAH ensures coordination between the overview and the spreadsheet: *e.g.*, zooming on the overview is reflected on the spreadsheet by displaying the spreadsheet data corresponding to the bin currently in focus in the overview. Finally, NOAH maintains contextual and historical information (Figure 1d and 1e) while displaying visual cues (Figure 1f and 1g) so that users don't lose context during exploration. The primary contributions of our work are as follows:


Home
e. Breadcrumb
Navigation History
Home > Chicago - Denver
Home > Nashville - New Orleans
Home > Ashv
d. History
city
Ashville - Boston (Rows: 1073)
Ashville 74
Austin 598
Boston 401
Chicago - Denver (Rows: 631)
Chicago 443
Denver 188
LA
Total Rows: 2337
Start Row No: 1706
End Row No: 4042
Nashville - New Orleans (Rows: 909)
Nashville 454
New Orleans 455
a. Overview
availability COUNTIF ,">60"
771.00
478.00
1864.00
b. Aggregate Column
f. Overview bin corresponding to the visible spreadsheet data is highlighted
g. Cells corresponding to navigation and aggregation attribute are highlighted
c. Spreadsheet

	A	B	C	D	E			
1	city	neighbourhood	type	price	availability			
2	Ashville	28704	Private room	60.0	332.0			
3	Ashville	28704	Entire home/apt	115.0	348.0			
			Private room	55.0	360.0	1	1	9/26/2015
			Entire home/apt	135.0	345.0	2	31	3/8/2016
			Private room	49.0	319.0	1	2	11/1/2015
			Entire home/apt	134.0	302.0	2	14	1/26/2016
			Private room	50.0	82.0	1	34	4/18/2016
9	Ashville	28801	Private room	59.0	321.0	2	138	4/10/2016
10	Ashville	28801	Private room	66.0	296.0	1	227	4/6/2016
11	Ashville	28801	Private room	77.0	339.0	1	97	4/7/2016
12	Ashville	28801	Entire home/apt	85.0	17.0	3	85	3/29/2016
13	Ashville	28801	Private room	90.0	365.0	1	3	4/11/2016
14	Ashville	28801	Entire home/apt	99.0	198.0	7	123	4/11/2016
15	Ashville	28801	Entire home/apt	110.0	21.0	1	1	4/11/2016
16	Ashville	28801	Entire home/apt	115.0	328.0	2	25	11/1/2015
17	Ashville	28801		125.0	312.0	2	46	4/17/2016
18	Ashville	28801		135.0	94.0	2	104	4/16/2016
19	Ashville	28801		150.0	139.0	1		
20	Ashville	28801		160.0	344.0	2		

Figure 1: The NOAH interface consisting of (a) a zoomable overview and (b) an aggregate column integrated with (c) a spreadsheet. A context bar consisting of (d) a navigation history displaying locations visited so far using the overview, and (e) a breadcrumb showing the current navigation path (*e.g.,* Home). (f) The user's current focus in the spreadsheet is highlighted on the overview. (g) Columns corresponding to the navigation attribute (city) and aggregate column (availability) are highlighted on the spreadsheet. See video demo at http://tiny.cc/NOAH.

- We introduce the notion of a *dynamic hierarchical overview* as an approach to eliminate cumbersome scrolling and steering during exploration on large spreadsheets.
- We identify a number of important *design considerations* for such an overview, as well as potential use cases.
- We abstract the problem of constructing our overview as one of *automatic hierarchical histogram construction*, and propose a solution for it. We also develop an interface to allow users to customize their hierarchies.
- We identify a *suite of novel interactions* for our overview—and determine how they can be synchronized with our spreadsheet; likewise, we determine how interactions on the spreadsheet can be synchronized with the overview.
- We introduce the notion of *aggregate columns* to support in-situ computation at various levels of the overview hierarchy without cumbersome steering.
- We implement these ideas in NOAH, *a plugin to* DATASPREAD, an open-source spreadsheet.
- We conduct a user study to evaluate the benefits and limitations of this plugin. The study required users to perform tasks that were representative of popular spreadsheet operations, using Microsoft Excel and NOAH. Compared to Excel, participants were able to complete spreadsheet navigation tasks more correctly (2.5× **fewer mistakes**) and quickly (2× **faster completion time**) using NOAH.

Our NOAH interface was demonstrated as part of an award-winning ICDE 2019 demo paper [13] and is available as part of the recent open-source DATASPREAD [1] release.

2 NOAH USE CASES

NOAH introduces new ways to explore spreadsheet data supported by interactions on a dynamic hierarchical overview; this overview in turn abstracts the raw spreadsheet data using high-level visual summaries. Table 1 captures the types of spreadsheet tasks for which NOAH enhances the user experience, drawn from an exhaustive list collated by Brehmer et al. [16]. We describe these tasks in the context of a real usage scenario below.

Purpose	Use Cases
Search	`browse` (*searching based on characteristics where location is unknown/known*, e.g., Rebecca tries to find Chicago listings with availability greater than 60 days).
	`locate/lookup` (*searching based on entities where location is unknown/known*, e.g., Rebecca wants to find all entries corresponding to a given city like Chicago).
Analyze	`identify` (*returning the characteristics of entity found during search*, e.g., Rebecca wants to examine Chicago listings to assess typical availabilities of listings in Chicago).
	`compare` (*returning characteristics of multiple entities*, e.g., Rebecca wants to compare listing patterns in Boston to that of Chicago).
	`summarize` (*returning characteristics of several entities*, e.g., Rebecca wants to gain an understanding of overall rental patterns across cities)
Produce	`generate/record` (*generation or recording of new information*, e.g., Rebecca issues an aggregate formula to generate summary availability statistics across cities)

Table 1: Use cases where NOAH enhances exploration experience.

Rebecca, a journalist, is exploring the *Inside Airbnb* dataset [29], a dataset of all the Airbnb listings across different US cities with ≈ 100*k* records and 15 attributes, within a spreadsheet. This dataset was created to investigate the long-standing accusation that many listings in Airbnb are illegally run as hotel businesses, while avoiding taxes; any listing available for rent for more than 60 days a year is considered to be operated as a hotel [51]. Given that this is the first time she's examining this dataset, Rebecca wants to first gain a bird's eye view of the data. Without NOAH, Rebecca would have had to use a pivot table to construct a summary. However, since this summary is disconnected from the underlying data and in a separate area of the spreadsheet, it is hard for Rebecca to map the summary statistics to the raw data to obtain further details about listings from any given city—she would have to switch back and forth between the pivot table results and the raw listings. With NOAH, she organizes the overview by city and starts casually exploring the dataset, understanding which cities are present, and roughly how many listings does each city have—with NOAH providing a high-level overview of cities (Figure 1a). We discuss the construction of the overview for arbitrary spreadsheets and associated interactions subsequently. The overview consists of sorted non-overlapping bins containing one or more cities. She can click on any bin and the corresponding data will be displayed at the top of her screen. For example, as shown in Figure 1c, clicking on the *Ashville-Boston* bin displays the Ashville listings (`locate`).

Next, say Rebecca wants to analyze one of the larger cities to understand the overall renting pattern (`summarize`). She decides to focus on Boston, her hometown, and wants to find out how many listings in Boston violate the "rent availability > 60 days" condition (`identify`). In a typical spreadsheet, Rebecca needs to manually steer and then select the Boston listings as input to a `COUNTIF` formula that counts the number of rows that satisfy the above mentioned condition. Using NOAH, she can zoom into the *Ashville-Boston* bin (Figure 2a and 2b) and then issue a `COUNTIF` operation on the overview. The result is displayed as an *aggregate column* alongside the overview (Figure 1b). With the raw data presented side-by-side, she can also dive into other attributes of the listings operating as hotels to see if there are any other identifying characteristics (`identify`). As she uses the overview to navigate to other cities, *e.g.*, Chicago, and compare the rent availability (`compare`), NOAH automatically updates the aggregate column results corresponding to that city (Chicago). In traditional spreadsheets, she would have to reissue the steering operation for each city being compared from scratch, which is cumbersome.

Finally, as Rebecca navigates the data, her navigation history (Figure 1d), *i.e.*, recently visited cities, and current navigation path (Figure 1e), is kept up-to-date, allowing her to maintain context during navigation. She can revisit any previously visited cities (`lookup`) by simply clicking on the relevant path in the navigation history. Maintaining the navigation history in traditional spreadsheets can be tedious as she has to manually create named-ranges. We explain this feature in Section 4.

3 DESIGN CONSIDERATIONS

In this section, we outline our design considerations for creating a *dynamic hierarchical overview* for spreadsheets aimed at eliminating cumbersome scrolling and steering during exploration. Our design considerations were informed by prior work on information visualization [86], overview+detail interfaces [28] and multiple-coordinated views [92] from human-computer interaction, and refined through our experiences across multiple design iterations.

DC1. Construct the overview in-situ. An overview helps users get a high-level picture of the data. However, maintaining the overview in a separate location from the data can lead to loss of context; instead, having it co-located with the data can help users make rapid glances to explore information between a bird's-eye view and close-up details [39].

DC2. Ensure reduced visual discontinuity while providing details-on-demand. Users often need to access subsets of data, and study their properties in detail, *e.g.*, via steering to issue formulae. Navigating back and forth between different subsets of data can lead to increased visual discontinuity. The interface should allow users to compute such details for various data subsets on demand [86]. The interface should maintain visual continuity as users navigate to a different subset, recomputing the details for the new subset.

DC3. Balance the screen space afforded to the overview. As the overview has limited screen-space available, we need to consider the trade-off between visual discontinuity (*DC2*) and clarity. Displaying a fine-grained overview improves visual clarity while increasing visual discontinuity—users need to scroll through the overview to access distant subsets of data. Displaying a coarse-grained overview decreases visual discontinuity at cost of reduced visual clarity—the overview may span too many data subsets and appear visually cluttered. The interface should further allow users to control the screen-space allocated to the overview.

DC4. Enable coordination between the spreadsheet and overview. Since users can view the overview and the spreadsheet simultaneously, interactions on both need to be linked [84], *i.e.*, an interaction on one should be reflected on the other [92]. For example, as a user scrolls through the spreadsheet, the user's current focus should be highlighted on the overview. However, not all interactions need to be interlinked, *e.g.*, changing the font size of a spreadsheet cell need not lead to a change in the overview.

DC5. Facilitate customization of the overview. As the overview is automatically generated, it may not reflect domain-specific context known only to the user [79]. For example, an overview constructed on a grading spreadsheet by binning nearby scores may not match the letter grade ranges that the instructors have in mind. Allowing users to customize the overview is therefore essential.

DC6. Display contextual and historical navigation information. The interface should record navigation history, allowing users to revisit previously visited locations [86], while also displaying their current navigation path for context.

4 DYNAMIC HIERARCHICAL OVERVIEWS

We now explain the design of dynamic hierarchical overviews as part of NOAH, the underlying algorithm for overview construction, and extensions to the design.

4.1 Overview Design

Spreadsheets often have one or more tabular regions containing structured data [9], interspersed with formulae—each tabular region essentially corresponds to an (ordered) relation. Our overview can be constructed on any of these regions on-demand. The overview is constructed in-situ (**DC1**), next to the spreadsheet, sorted and organized by an attribute of this region called the *navigation attribute*, selected by the user. (Sorting is a commonly used operation within spreadsheets.) Any attribute type that can be ordered can be a navigation attribute, *e.g.*, text, numbers. The overview is constructed at multiple granularities hierarchically. Each granularity is divided into non-overlapping groups of data called *bins*. As shown in Figure 2d, an overview of the Airbnb data on the navigation attribute "city" has granularity levels. The highest (coarsest) granularity level consists of four bins. Figure 2a depicts the first four bins, the first of which is *Ashville-Boston*. Each bin contains summary information regarding the data subset/region it spans, *e.g.*, starting row and ending row number, and the total number of rows the region spans. Each bin displays an overview of the next (finer) granularity (if any) with embedded bar charts. For example, in Figure 2d, the topmost bin (*Ash-Bos*) spans three cities (*Ashville*, *Austin*, *Boston*), each of which is a bin in the next (finer) granularity. Correspondingly, Figure 2a shows three horizontal bar charts for the first *Ash-Bos* bin, one for each bin in the next granularity. Since the third bin from the top (*LA*) spans only one city, no bar chart is embedded. Users can perform different operations on the bins, *e.g.*, clicking to pan and zooming in/out. NOAH supports other interactions atop this multi-granularity overview, *e.g.*, customization and aggregation. We discuss these interactions in Section 4.3.

4.1.1 Overview Spreadsheet Coordination. NOAH supports coordination between the dynamic hierarchical overview and the corresponding spreadsheet (**DC4**), *i.e.*, interactions on the overview may be reflected on the spreadsheet and vice-versa. One example of this coordination is indicating the navigation attribute on the spreadsheet using color (see the lime green column in Figure 1c) as user constructs the overview. However, not all overview interactions are coupled with the spreadsheet and vice versa. The coupling depends on the user's current focus—*to ensure consistency between the overview and the spreadsheet, any interaction on either interface that changes the current focus must be reflected on the other interface.*

4.1.2 The Rationale for Binning. A conventional design for overviews within popular interfaces is as a spatially partitioned collection of thumbnails on the left of the standard detailed view, similar to Microsoft Power Point or Adobe Reader. However, displaying too many thumbnails results in increased scrolling to access distant thumbnails, increasing visual discontinuity. On the other hand, displaying too few thumbnails reduces visual discontinuity, but at the cost of visual clarity—the thumbnails appear cluttered and fail to represent the underlying data clearly [28]. To strike a balance between these two objectives (**DC3**) we designed an overview that abstracts the data at varying levels of detail, in a hierarchical or multi-granularity fashion. Multi-granularity representations have been shown to work better for larger datasets—presenting information at multiple granularities makes visual representations more perceptually scalable and less cluttered [33]. Thus, this hierarchical overview provides an alternative to the conventional spatially partitioned single-granularity representation of the data space, *e.g.*, in Power Point, by allowing users to control the scale at which the overview should be displayed [28]. Users can resize the overview to control the amount of spreadsheet data that remains visible.

The data structure underlying the overview is a histogram constructed on the values in the navigation attribute column. We describe the formal problem in the next section. Histograms result from *binned aggregation*—consecutive records or rows are grouped into bins (or groups), where each bin represents a group of rows and is associated with a `COUNT` aggregate, capturing the number of rows that fall in that group. Unlike a traditional `GROUP BY` where each distinct value is a separate group, here, multiple consecutive distinct values can form a group. In addition to providing high level (*e.g.*, densities) and low level (e.g., outlier) details, binned aggregation techniques enable a multi-granularity visual representation of data by varying the bin size and have therefore been used in interactive visualization of large datasets, *e.g.*, in *imMens* [58]. An additional benefit of a binned overview for spreadsheets is a decrease in visual discontinuity during navigation. As users are able to view an overview that fits in the computer screen, they can quickly navigate the data—the bins act as landmarks in the overview, enabling users to skip irrelevant bins and quickly navigate to the desired subset of data. We now discuss how the overview is constructed.

4.2 Overview Construction

A dynamic hierarchical overview is constructed on a given tabular region within a spreadsheet. We discuss extensions to support multiple tabular regions in Section 4.5. Since current spreadsheet systems only support a few million rows, the overview construction is performed in-memory and is extremely fast.

4.2.1 Problem formulation. Suppose we have a tabular region T with n rows, with a set of columns $\mathcal{A}$. We assume T is static for now, and consider edits in Section 4.5. When the user request a dynamic hierarchical overview on T, they designate a special *navigation attribute* (column) $A \in \mathcal{A}$. The tabular region T is then sorted by this attribute; any attribute can be sorted this way and can lead to a meaningful overview, including text, numbers, and dates. We can also support navigation across multiple attributes; we discuss this extension in later on. Across rows in T, the values taken on by A are $v_1 \leq v_2 \leq \ldots \leq v_n$: *i.e.*, v_i is the value of A in the ith row. For example, in the usage scenario explained earlier, v_i corresponds to city names across rows, which can be ordered lexicographically. We now focus on generating the first level of our dynamic hierarchical overview as an *approximately equi-depth histogram* on column A; we extend this technique to a hierarchy in Section 4.2.3. First, a histogram can be defined as follows.

Definition 1 (Histogram). *A k-histogram on values $v_1 \leq v_2 \leq \ldots \leq v_n$ of A is defined by* split points $s_0 = -\infty < s_1 < s_2 < \ldots < s_{k-1} < s_k = \infty$ *such that all v_i where $s_{j-1} < v_i \leq s_j$ are said to be* assigned *to the jth bin of the histogram, B_j.*

Thus, a k-histogram induces a *partition* of the v_i into k bins $B_1, \ldots, B_k$. We indicate the *size* of the bin B_i, $|B_i| = b_i$ to be the number of records assigned to that bin. Here, we are partitioning the rows of our spreadsheet into k coarse-grained bins to be displayed as part of our overview. Intuitively, we want these bins to be *balanced*, such that each bin has the same number of v_i (and therefore records) assigned to it. One way to ensure that is via *equi-depth histograms.* Equi-depth histograms are commonly used for summarizing statistical properties of data, with applications in query optimization and approximate query processing in database systems [45], distribution fitting in data streams [67], and interactive scrolling on large spreadsheets [80]. We define an equi-depth histogram constructed on A as follows:

Definition 2 (Equi-Depth Histogram). *A k-histogram is said to be an* equi-depth histogram *if* $b_j = \frac{n}{k}, \forall j \in [1, k]$.

In the scenario where the values v_i are all distinct, it is easy to find an exact equi-depth histogram (modulo rounding errors). However, it is more typical that A has repeated values among the v_i. For example, there are multiple listings per city in the Airbnb dataset. In such a scenario, it is unlikely that we will find split points s_i corresponding to an equi-depth histogram. We therefore introduce the problem of discovering of an *approximately equi-depth histogram.* We denote $\hat{b} = \frac{n}{k}$ to be the expected size of each bin B_j if we could construct an equi-depth histogram. Here, we want each b_i to be as close to $\hat{b}$ as possible.

Problem 1 (Approximately Equi-Depth Histogram). *Given a tabular region T, a spreadsheet navigation attribute A, and the number of bins k, return a k-histogram with split points $s_1, s_2, \ldots, s_{k-1}$ yielding bins $B_1, B_2, \ldots, B_k$ such that* $\max_j |b_j - \hat{b}|$ *is minimized.*

Thus, we want to ensure that the maximum distance of any b_i from $\hat{b}$ is minimized.

4.2.2 Dynamic Programming Algorithm. To solve Problem 1, we can use a dynamic programming algorithm. One approach would be to try to come up with the best way to bin values $v_1, \ldots, v_i$, for $i \in [1, n]$ into j bins, for $j \in [1, k]$ via a recurrence relation expressed using sub-problems. However, many of these binnings would not lead to a valid histogram, since multiple consecutive v_i that have the same value may be spread across two or more bins, giving us no valid split points as is mandated by Definition 1. So instead, we operate on the distinct values across the v_i, we let these be $u_1 < u_2 \ldots < u_m$. We let c_i be the cardinality of u_i, i.e., the number of v_j such that $v_j = u_i$. Then, we have the following recurrence relation, for all $i \in [0, m]$, $j \in [0, k]$

$$H(i,j) = \begin{cases} 0, \text{if } i = 0 \text{ or } j = 0 \text{ or } i < j \\ \min\limits_{1 \le d \le i} \max(H(i-d, j-1), |\sum_{l=i-d+1}^{l=i} c_l - \hat{b}|), \text{otherwise} \end{cases}$$

At a high level $H(i, j)$ encodes the best way to partition the first i distinct values of v, i.e., $u_1, \ldots, u_i$, across j bins. While computing H, we can also record the split points, allowing us to reconstruct the histogram. The complexity of this algorithm is $O(m^2k)$. Typically $m \ll n$, so this algorithm is fairly fast.

In NOAH, we adopt a simple "best effort" greedy algorithm to construct this approximately equi-depth histogram in $O(m)$. We consider the distinct values u_i in increasing order, one at a time, and keep greedily adding bins, one at a time, until we have assembled enough u_i to justify adding a new bin. Formally, say we have added $i - 1$ bins so far, using distinct values up to and including u_{j-1}: that is, all rows with $v_i \le u_{j-1}$ have been assigned to a bin. We construct the ith bin using values $u_j, u_{j+1}, \ldots$, until we hit the cutoff that $\sum_{l=1}^{i} b_l - i \cdot \hat{b} > 0$. Once we pass this cutoff, this means that we have surpassed the average size $\hat{b}$ for the first i bins, and the remaining u_j must be assigned to bins $i + 1, i + 2, \ldots$ onwards.

4.2.3 Extending to a Hierarchy. Our dynamic hierarchical overview is constructed on-demand in a top-down fashion (see Figure 2d). At the start, we only compute the highest level ($l = 1$) of the overview consisting of an approximate equi-depth histogram with k bins; these k bins are displayed. If the user drills down or zooms into into any of these k bins (using actions that we will discuss in the next section), say bin B_i, we initiate a further subdivision of the records assigned to B_i into k bins once again via approximate equi-depth histogram construction. Doing so for each of the k bins at level $l = 1$ gives us k new k-histograms at level $l = 2$. In this manner, conceptually, we have a tree of fanout k constructed on-demand as the user explores the tabular region. Note that if the number of distinct values associated with a bin B_i, m, is less than k, then we display fewer than k bins on drilling into B_i—each such bin will correspond to a single distinct value. At an extreme, if the number m of distinct values associated with a bin B_i is 1 (i.e., all records in that bin have the same value v_j for attribute A), then we do not permit drilling down into that bin, and that bin will end up not having any children in our hierarchy.

While we construct our hierarchy lazily, we materialize the split points and row numbers for any nodes (i.e., bins) in the hierarchy that have been visited in the past. This way, we avoid repeated histogram construction, and make drilling or zooming in/out of the hierarchy more responsive, at the cost of a little additional space.

It might not have escaped the reader that the hierarchical overview we are constructing has parallels to B+-trees in that our hierarchy has a fixed k-way fanout. There are several key differences, however. First, our hierarchy need not be fully balanced, unlike B+-trees, especially when certain distinct values have high cardinalities. Bins containing those values may have fewer than k children—and in the extreme, may have no children. Moreover, to guarantee efficient retrieval and maintenance, B+-trees impose a constraint on the number of keys (children) per internal node which cannot be violated—our overview has no such restriction. Furthermore, as we discuss next, users can customize the bins which may result in the constraint on the number of keys being violated. Finally, our goal with an approximate equi-depth histogram at each level of the hierarchy was to ensure a near-equal subdivision of records, making it easier for human consumption. B+-trees are not meant for human consumption, and are simply meant for efficient access.

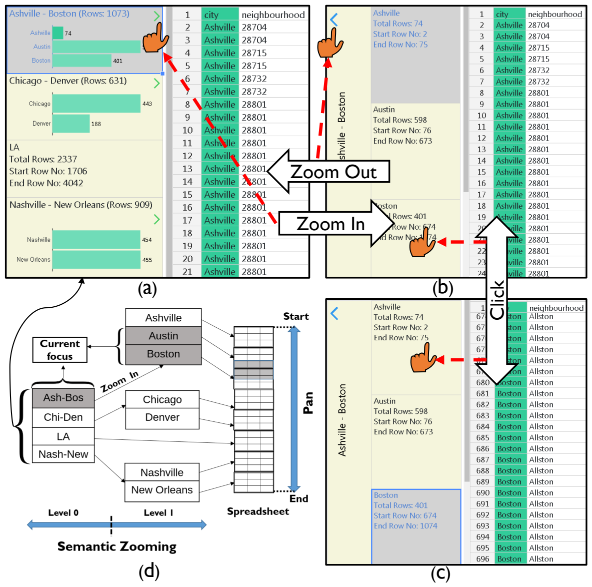


Figure 2: Navigational operations. (a) The overview at the highest level of granularity. (b) A zoomed in view of the *Ashville-Boston* bin. (c) As the user clicks on the *Boston* bin, the Boston listings are displayed on the sheet. The *Boston* bin is highlighted in gray to indicate user's current focus. (d) Conceptualizing the hierarchical overview.

4.3 Operations and Interactions

We now discuss the operations and interactions that can be performed on the overview.

4.3.1 Clicking and Semantic Zooming. Clicking a bin is an example of an interaction that is coupled between the spreadsheet and the overview (see Section 4.1.1). When a user clicks on a specific bin, NOAH displays the corresponding spreadsheet data. Users can use this operation to jump to a specific spreadsheet location using the overview without having to scroll endlessly. For example, in Figure 2b, as the user clicks on the *Boston* bin, the data corresponding to Boston is displayed (Figure 2c). Conversely, as the user scrolls up on the spreadsheet, both Austin and Boston listings appear in the screen. As the current focus on the spreadsheet changes, both the Austin and Boston overview bins are highlighted (see Figure 2d). Note that the click operation is different from the traditional spreadsheet `Filter` operation. `Filter` hides spreadsheet data that do not satisfy the filtering condition while clicking brings the desired subset of data in view without hiding the rest.

Users can zoom into a bin (and thereby descending to a lower level of the hierarchy) to view more fine-grained information or zoom out to view more coarse-grained information, via what is known as *semantic zooming* [75]. The zooming operation is decoupled—when a user zooms into a bin already in the user's current focus, the spreadsheet view does not change. For example, in Figure 2a, from the bin *Ashville-Boston* when the user zooms in to the next level, NOAH displays the bins *Ashville*, *Austin*, and *Boston* (Figure 2b); here, the spreadsheet view stays the same. If the user zooms out of the current granularity, again NOAH displays the bins *Ashville-Boston*, *Chicago-Denver*, and others. The zoom out operation is also decoupled—when a user zooms out, the overview displays a coarser granularity view of the user's current focus. As explained in Section 4.2.2, users can only zoom into any bin that contains multiple unique values. For example, in Figure 2d, at level 2, each bin corresponds to one city. Users can only click on those bins to bring that data into view, and cannot zoom in further.

4.3.2 Overview Customization. As NOAH constructs the hierarchical overview automatically, the overview binning may not capture domain-specific context or user needs. NOAH enables users to customize this organization (**DC5**). At any granularity, users can merge multiple consecutive bins into a single bin, or split a bin into multiple bins. The other operations include merging all the bins into one single bin and collapsing all the bins into singular bins, *i.e.*, one distinct value per bin. Say the user wants to compare summary statistics of Boston and Chicago. In the current organization these two cities are in two different bins (see Figure 3a). Using bin customization, the user can merge the two bins *Ashville-Boston* and *Chicago-Denver* to create a new bin *Ashville-Denver*. Users can then zoom into this bin and compare summary statistics of the cities in the same view. The interactions for splitting a bin depend on the data type. If the navigation attribute is textual, any bin can be split into as many bins as the number of distinct values that bin contains. If the navigation attribute is numeric, users can split the bin into an arbitrary number of bins. NOAH does not allow users to rearrange the order of the bins: since the overview represents a histogram, the bins are ordered; reshuffling the bins violates that order. Users can only customize the boundary of the bins.

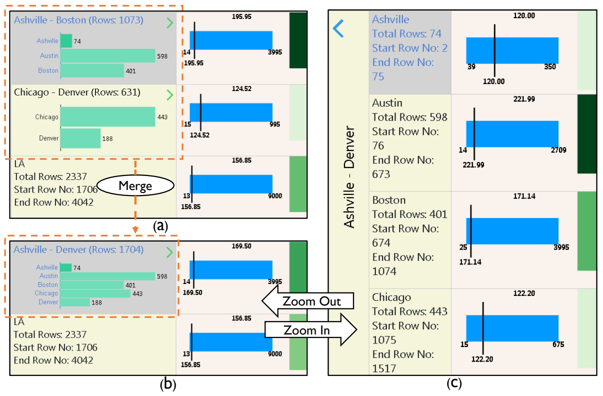


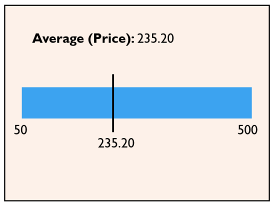
Figure 3: (a) Chart view of the aggregate column. (b) A new bin is created by merging the top two bins. (c) Zooming into the newly created bin.

4.4 Aggregate Columns

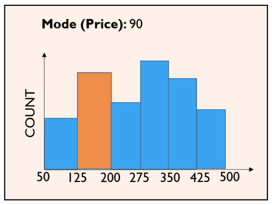
Users can issue spreadsheet formulae directly on the overview to compute aggregates for the data in each bin. This feature allows users to not have to perform cumbersome steering operations to issue formulae—with the formulae having been declared once as part of the overview, and dynamically updated as the users traverse up and down the hierarchy. The results are displayed as an *aggregate column* (see Figure 1b). This operation is similar to `GROUP BY` queries where user can select one or more *aggregates* to apply on measure attributes, such as `SUM`, `COUNT`, or `AVERAGE`. However, unlike `GROUP BY` queries where each group corresponds to one distinct value u_i, in NOAH, several consecutive distinct values may form a group depending on the overview granularity. For example, the top two results in the aggregate column (denoted with a blue bar) in Figure 3a correspond to multiple cities each as a group while the bottom result corresponds to a single city, LA. We now discuss how aggregate columns are created and how the corresponding results are materialized and displayed.

4.4.1 On demand Column Creation. Creating an aggregate column is equivalent to selecting subsets of data on the sheet, *i.e.*, steering, and then executing a formula on this subset. Users are not required to explicitly type formulae; rather they simply select the formula from a drop-down menu, and provide the necessary formula parameters to a form. Users can issue several formulae simultaneously, each creating a new aggregate column. The aggregate column can employ any statistical or mathematical formulae that operate over a range of data. However, adding an aggregate column takes up screen space, shrinking the spreadsheet view. As a workaround, users can resize or remove aggregate columns if required (**DC3**). When the user issues a formula on the overview, the spreadsheet column corresponding to the aggregate column is highlighted in grayish orange (see Figure 1c)—another example of coupled interaction. For conditional formulae like `COUNTIF`, cells that satisfy the condition are highlighted, *e.g.*, in Figure 1c, the cells with availability ≥ 60 are colored in sky blue. In this manner, users can quickly determine which cells are relevant to the aggregation operation. When a user zooms into a bin for the first time, the overview is updated and the aggregate column is computed. The aggregates across the hierarchy are reminiscent of drill-downs or roll-ups in data cubes [38]; here, we are drilling down/rolling up on automatically grouped values of a given attribute rather than across multiple attributes.

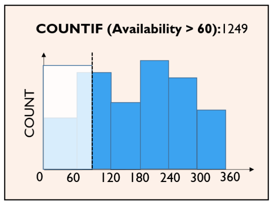
4.4.2 Multi-perspective Result Representation. Users can view the results either as raw values (see Figure 1b) or as charts (see Figure 3), and can toggle between the two representations. Raw values are displayed along with a colored bar, the *value bar*, whose length is proportional to the corresponding aggregate (see Figure 1b). Users can use the lengths to visually compare across bins. The chart representation varies depending on the formula type. We have classified the formulae supported into five categories: a) summary (*e.g.*, min, max, average), b) frequency (*e.g.*, mode, large, small), c) conditional (*e.g.*, countif, sumif), d) spread (*e.g.*, var, stdev), and e) others (*e.g.*, sum, count). All other categories except for the *others* category can be represented by charts. Figure 4 shows the chart representation for these categories along with different visual cues that highlight formula results as well as other information. For example, for *conditional* formulae like `COUNTIF`, shading is used to de-emphasize



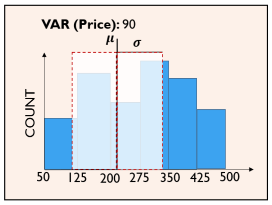
(a) Chart Type: Summary
Representation: Horizontal bar
Visual Cue: Vertical line (result)



(b) Chart Type: Frequency
Representation: Histogram
Visual Cue: Color (highlight)



(c) Chart Type: Conditional
Representation: Histogram
Visual Cue: Vertical line (result),
Shade (de-emphasize)



(d) Chart Type: Spread
Representation: Histogram
Visual Cue: Vertical line (mean),
Shade (variance)

Figure 4: Formula types and chart representations.

data ranges that do not satisfy the condition. A similar shading technique is used for the *spread* category to highlight the standard deviation. For the *frequency* category, coloring is used to identify the bin with the result, *e.g.*, mode. In the chart representation mode, each entry of the aggregate column contains one additional visual cue—a color bar with shades of green on the right of the chart (see Figure 3). The darker the color, the higher the value corresponding to that entry. Users can utilize the color intensity to compare the results among different aggregate column entries.

4.4.3 Constructing the Aggregate Column. The aggregate column is kept in sync with the bins as users zoom in and out, eliminating repeated steering operations. The aggregate column entries are materialized alongside the corresponding histogram bin, ensuring interactivity as user performs ad-hoc navigation via clicking or zooming. The user can also perform ad-hoc navigation from the context bar—enabling them to explore previously visited overview bins (**DC6**). The context bar consists of two components: a) a breadcrumb displaying the current navigation path (see Figure 1e), and b) a navigation history maintaining a list of recently visited bins (see Figure 1d). As the user explores previously visited bins using these two features, the corresponding aggregate column results that were previously materialized are displayed, providing interactive response times. Even when visiting a bin for the first time, constructing the aggregate column simply requires a single scan over the already sorted data, which is fairly efficient.

4.5 Extensions

We now discuss how our techniques can be extended to address a wider range of scenarios than those discussed previously.

Multiple Tabular Regions. In Section 4.2, we discussed how our dynamic hierarchical overview can be constructed for a single table or tabular region. However, spreadsheets can have multiple tables, as well as formulae and text interspersed. In such cases, NOAH can support exploration for each tabular region independently via a corresponding hierarchical overview, supported via an overall map-like overview for users to select which tabular region they want to explore in detail. We can leverage existing work on spreadsheet table detection and structure identification [6, 21–24, 31] to identify such regions to construct this overall map.

Overview Maintenance. Spreadsheet users often perform various edit operations, *e.g.*, updating values, adding/removing rows or columns. So far, we assumed our data to be read-only, but we can extend our techniques to support edits. As a trivial case, when the spreadsheet cell being edited does not pertain to the navigation attribute or an attribute used in an aggregate column, we do not need to update the overview. When an attribute referred to in an aggregate column is deleted, the corresponding aggregate column is removed from the overview; deletion of the navigation attribute results in the removal of the entire overview. We consider other fine-grained cases next.

When edits are performed to a cell pertaining to the aggregate column, the materialized results for that column need to be updated. We simply perform incremental view-maintenance-style delta updates to the corresponding bin at the lowest level of the hierarchy with materialized aggregates for the updated aggregate column, and percolate the same changes up the hierarchy. This update can be done in time proportional to the height of the hierarchical overview. Likewise, when cell-level edits are performed to the navigation attribute, the hierarchy needs to be updated. Since the tabular region is meant to be in sorted order of the navigation attribute, we will move the updated row to its new position in the sheet (which can be handled in $O(\log n)$ using positional indexes [9], discussed more in the next section); we then need to update the number of records in each materialized bin that may be impacted by this change. The latter requires two traversals up the hierarchy one starting from where the record was (decrementing counts), and one where it was moved to (incrementing counts). For cell-level edits, we do not recompute the binning or hierarchy construction, nor do we update the split points—since such a drastic change may be more jarring than useful for the end-user. There are certain corner cases that we need to handle more delicately, such as when a cell-level edit results in a distinct value within the navigation attribute being removed entirely, or when a new distinct value is introduced. For both these updates, we do not modify the hierarchy unless absolutely needed, *e.g.*, if a bin suddenly becomes empty, or if a bin that had only a single distinct value now has multiple values (requiring a potential drill-down). In all cases, a cell-level update to a navigation attribute also leads to an update of all of the aggregate column entries for each of the bins that are updated. Row additions and deletions are handled similarly to cell-level updates. For large-scale changes (*e.g.*, a large fraction of the rows are moved or deleted), we can default to recomputing the overview from scratch.

Supporting Dynamic Crossfiltering. Currently, the charts displayed in an *aggregate column* are non-interactive (see Figure 4), *i.e.*, users cannot interact with the charts to visually look up relevant or interesting data points within the spreadsheet. Such multi-modal linked selections is similar to crossfilter [66, 101] where one interaction may generate hundreds of queries/formulae per second. For example, for the *conditional* chart representation in Figure 4c, a natural liked interaction is to filter and display spreadsheet rows that satisfy the "Availability" condition in the `COUNTIF` formula. Sliding the vertical line left or right will update the condition and the relevant cells to be filtered needs to change accordingly. As discussed in Section 4.1, each bin in NOAH overview maintains the start and end row index of the spreadsheet data corresponding to that bin. Therefore, NOAH can quickly scan the relevant row ranges in memory and filter out the relevant cells while leveraging the formula engine to compute the newly formulated formula results to be displayed in the aggregate column.

Enabling Flexible Overview Binning. The experience surrounding the construction of the overview can be further improved, especially for categorical data. Currently, the bins of the overview can be customized only after the overview is constructed. Providing the users the ability to specify the binning criteria via UDFs is another possible extension. For example, users can plug-in an agglomerative clustering algorithm to construct a similar hierarchial overview.

Multiple Navigation Attributes. We finally note that our hierarchy can also be constructed for multiple attributes (*e.g.*, explore the Airbnb data by city and neighborhood). One simple way to do so would be to consider the values of the two attributes to be treated together as a pair, sorted by the first attribute first, and then the second. There is one small wrinkle in that some of the histograms may not be interpretable if they subdivide based on both attributes—so we additionally make the restriction that a histogram is only constructed on one of two attributes at a time. This restriction has the effect of having the higher levels of the hierarchy based on the first attribute, and lower levels based on the second attribute.

5 SYSTEM ARCHITECTURE

We have integrated NOAH with DataSpread [9], a web-based spreadsheet, as a plug-in. Since NOAH is a plug-in, it has a separate front-end and back-end that communicate and coordinate with the DataSpread front-end and back-end respectively. The NOAH front-end is built with HTML/CSS/JavaScript technologies along with D3 [14] for generating charts.

System Design. We depict the architecture diagram in Figure 5. On the client or front-end side, the NOAH client is responsible for capturing user input on the overview (clicking on bins, history, or the context bar, creating an aggregate column or customizing the binning, zooming in/out), and for rendering the hierarchical overview and its components (*e.g.*, the aggregate column, context bar) based on results returned by the NOAH back-end. Given any user interaction on the front-end, the NOAH *request processor* issues a request to the back-end NOAH *controller*. This controller propagates requests to the appropriate manager. The *hierarchy* and *aggregation* managers are responsible for the creation and maintenance of the hierarchy and aggregate column(s) respectively. We discuss how these managers leverage DataSpread's computation engines and access methods subsequently. The *navigation and synchronization manager* is responsible for the management of the user's navigation context and history, *i.e.*, deciding which bin to highlight, and coordinating with the DataSpread *sheet controller*. The *navigation and synchronization manager* notifies the *sheet controller* to update the visible spreadsheet data based on user's interaction on the overview. Conversely, the *sheet controller* notifies the *navigation and synchronization manager* to update the current overview context based on user's interaction on the spreadsheet. The NOAH backend also maintains the current materialized hierarchy, the materialized aggregation results, as well as the history.

Integration with DataSpread. DataSpread also has front-end and back-end components, tracking user actions and displaying the sheet, and performing computation and retrieving data on demand, respectively. DataSpread uses positional indexes [9], an indexing structure that allows for positional access (*i.e.*, retrieving data by row number) and updates (*i.e.*, adding/deleting rows), both in $(\log n)$. The index is essentially a *counted* B+-tree wherein the number of records under each sub-tree is recorded and maintained. The NOAH back-end leverages the positional index and formula engine to construct the hierarchical overview and aggregate column(s), respectively. For a given navigation attribute, *e.g.*, "city", when the data is first sorted by this attribute, the positional index is updated by DataSpread to reflect the new ordering. NOAH leverages this index to retrieve the entire collection of (*row id, attribute value*) pairs to construct an approximately equi-depth histogram (discussed in Section 4.2.2) on the attribute values, *e.g.*, city names, for the first level of the overview. As the data is sorted by the navigation attribute value and each value is associated with a *row id*, each bin in the overview can be mapped to a sequence of *row id*s. When users click on a bin, NOAH uses this mapping to display the corresponding spreadsheet rows.

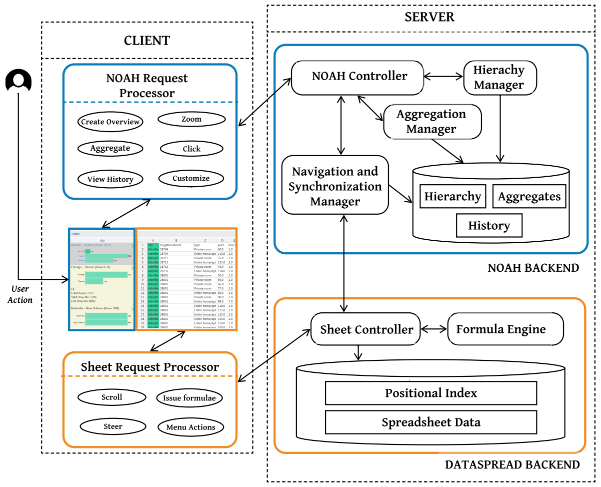


Figure 5: System architecture.

Integration with Other Spreadsheet Systems. NOAH can also be potentially integrated with existing spreadsheet systems like Excel and Google Sheets, since these systems all support access to cells by position, as is needed for formula computation. NOAH can be made more efficient if it can leverage the internal position/order management data structures within the spreadsheet—but this is not essential, except for performance. Since NOAH is a plug-in to spreadsheets, NOAH doesn't require any changes to the core spreadsheet internals, and can be engineered to act externally to the sheet, while still reading spreadsheet data and triggering spreadsheet computation. Instead, implementation of external coordination modules are required to synchronize operations on NOAH with the sheet; for example, a click on the overview should trigger a scroll in the sheet. Scrolling on the spreadsheet should not only display new data but also update the bin highlighted in the overview. Thus, event handlers need to monitor actions on the spreadsheet and trigger corresponding actions in the overview.

6 EVALUATION STUDY DESIGN

In this section, we present the design of a user study aimed at understanding the impact of our dynamic hierarchical overview while performing navigation and computation on spreadsheet data. Existing spreadsheet systems do not employ specific plug-ins for exploration. Since there is no equivalent system for comparison, we studied how participants' spreadsheet exploration experience vary with the presence or absence of such a plug-in. That is, we compared a NOAH-integrated spreadsheet system with a popular spreadsheet system, Microsoft Excel, across various tasks. These tasks were representative of the use cases presented in Table 1. Similar domain specific-evaluations have been performed for evaluating various overview+detail interfaces, *e.g.*, database browsing [70] or tree navigation [7]. Our study was designed to answer the following questions: RQ1) How does the integration of a dynamic hierarchical overview like NOAH impact the usability and efficiency of spreadsheet data exploration? and RQ2) How do the various components of NOAH impact users' spreadsheet exploration experiences?

6.1 Study Design and Participants

The two systems used in the study were: Microsoft Excel, and NOAH integrated as a plug-in within DataSpread [11] (henceforth, referred to as NOAH). We chose Excel for our comparative study because it is the most popular spreadsheet in use today; we chose DataSpread because the source code was open sourced allowing an easy integration. We recruited 20 participants (11 female, 9 male) via a university email newsletter. To ensure that prior experience with spreadsheets didn't affect the performance of participants during the quiz phase, we only recruited participants who rated their experience with Excel to be greater than four on a scale of one (no expertise at all) to five (very experienced) while signing up for the study. All of the participants were familiar with the basic mathematical and statistical functions supported by Excel. (Participants were free to use any Excel capabilities, including subtotals, pivot tables, and named ranges that partially address some of the data exploration concerns in conventional spreadsheets.) The study consisted of three phases: (a) an introductory phase to help participants familiarize themselves with NOAH, (b) a quiz phase where the participants used both the systems to perform targeted tasks on two different datasets (described later), and (c) a semi-structured interview and surveys to collect qualitative feedback regarding the quiz phase. The study protocol along with the list of tasks, surveys, and interview questionnaires are included in the supplementary material.

6.2 Datasets

We used two datasets—the birdstrikes [99, 101], and the Airbnb [29] datasets. These datasets were chosen for their understandability to a general audience. The birdstrikes dataset records instances of birds hitting aeroplanes in different US states. The dataset has 10,868 records and 14 attributes (eight categorical, one spatial region, one temporal, four numeric). The Airbnb dataset was larger than the birdstrikes dataset. To enforce parity among datasets, we created a sampled version of the original Airbnb dataset with 10,925 records, by uniformly sampling 10% of the records from each US city. This dataset has 15 attributes (six categorical, two spatial region, one temporal, six numeric).

6.3 Study Procedure

We now explain each of the phases of our study.

Phase 1: Introduction to NOAH. We began the study by showing a six-minute video tutorial explaining the features of NOAH on a dataset of all the flights across the US for January 2018 [73]. The participants then explored the same dataset using NOAH to familiarize themselves with the system for about 10 minutes.

Phase 2: The Quiz Phase. The purpose of the quiz phase was to evaluate the effectiveness of NOAH in addressing spreadsheet exploration limitations. Each participant performed specific tasks on the two datasets in two sessions, using Excel for one and NOAH for the other. We alternated the order of the datasets between consecutive participants. The order of the systems was alternated between every two participants. Each session was followed by a survey, described later. We developed an online JavaScript-based quiz system that recorded user responses and submission times. We also recorded the participants' interactions with both systems using screen capture software.

Category	Question (Q), Purpose (P), Use case (U)
steer	**Q**: Organize the data by State. How many flights that had damages (damage = 1) originated from Florida?, **P**: *Search*→*Query*, **U**: `lookup→identify`
find	**Q**: How many flights in the currently visible spreadsheet window have damages?, **P**: *Search*, **U**: `browse`
steer	**Q**: Organize the data by State. How many flights that had damages (damage = 1) originated from California?, **P**: *Search*→*Query*, **U**: `lookup→identify`
compare (2)	**Q**: Which state between Florida and California has a higher number of flights with damages?, **P**: *Query*, **U**: `compare`
compare (*N*)	**Q**: Find the state with the most birdstrike occurrences, **P**: *Query*→*Search*, **U**: `summarize→locate`
customize	**Q**: Organize the data by *altitude*. What is the average cost of damages for altitude bin 0-450?, **P**: *Query*→*Search*→*Produce*, **U**: `generate→summarize→lookup`

Table 2: Quiz tasks for the birdstrikes dataset. The task purposes and use cases correspond to Table 1.

Survey. After each session, participants rated the corresponding system on six metrics: confidence, comprehensibility, level of satisfaction, ease and speed of use, and ease of learning for spreadsheet exploration, on a Likert scale from one (*e.g.*, strongly disagree) to seven (*e.g.*, strongly agree).

Quiz Phase Tasks. For each dataset, we designed six tasks across five categories: steer (two tasks), find (one task), compare (2) (one task), compare (*N*) (one task), and customize (one task). These tasks encompass all the use cases underlying the *Search*, *Query*, and *Produce* purposes: `lookup`/`locate`, `identify`, `browse`, `compare`, `summarize`, and `generate`, from Table 1. These tasks were selected to mimic a typical spreadsheet exploration workflow and are representative of the most frequently issued spreadsheet operations [15, 55]. The tasks were presented in the same order as shown in Table 2 for the birdstrikes dataset. The tasks for the Airbnb dataset mimic a scenario similar to the example in the usage scenario section.

Evaluation. We evaluated the accuracy and completion time for all of the tasks. We combined this analysis with the findings from a qualitative survey, interview, and screen/audio recording data to provide insights that can be corroborated across multiple sources. Moreover, we analyzed the survey responses to quantify the usability of both the systems. We then measured the statistical significance of the comparisons between the two systems.

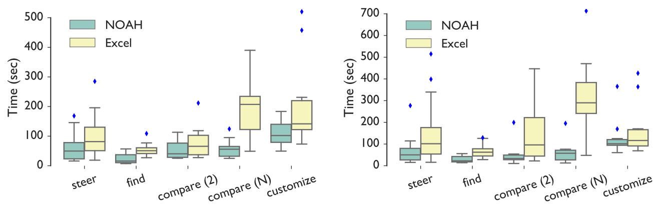
Phase 3: Interview Phase. We conducted a semi-structured interview to identify participants' preferred systems for different tasks and to understand the reasoning behind their choices. We also asked participants to comment on the usefulness of features present in NOAH and Excel.

7 RESULTS

In this section, we analyze the quantitative and qualitative data collected during the quiz and interview phases to address our research questions.

7.1 Exploration Performance and Usability

To answer RQ1, we first compare task completion times and accuracies in NOAH and Excel and then analyze the survey responses that evaluate the usability of the tools.

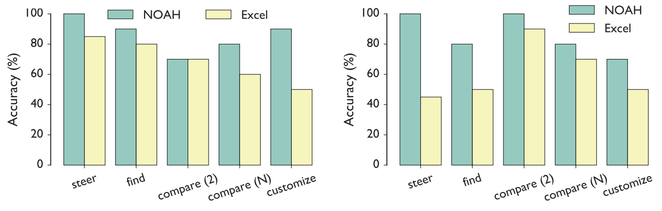


(a) birdstrikes **(b) airbnb**

Figure 6: Submission times per category. Median submission times are much smaller for NOAH compared to Excel.

7.1.1 *Faster exploration without sacrificing accuracy.*

In Figure 6a and 6b, we show the distribution of submission times of participants for the five task categories, for birdstrikes and Airbnb respectively. For most categories, *participants' median submission times using NOAH were less than the fastest submission times using Excel.* This observation suggests that the capabilities offered by NOAH made spreadsheet exploration faster for these tasks. The majority of submission times using NOAH were faster than Excel—19 out of the 20 participants completed at least four tasks in less time using NOAH.



(a) birdstrikes **(b) airbnb**

Figure 7: Per category accuracy indicates higher accuracy of task completion in NOAH compared to Excel.

In Figure 7a and 7b, we show the percentage of correct submissions for the four quiz task categories, for birdstrikes and Airbnb, respectively. For all the tasks except for the fourth task, compare (2), for which the accuracy was the same for both tools, participants attained slightly higher accuracy with NOAH compared to Excel. We discuss the accuracy of the fourth task later in the section.

Statistical Significance. We further evaluated the statistical significance of the task performance results, *i.e.*, completion time and accuracy. To measure the significance of the task completion times, we ran *Mann-Whitney's U test* [60] (as completion times did not follow a normal distribution). *For all of the tasks except the customize task, we found a significant effect of the tools, i.e., the response times for the tasks significantly differed by the choice of the tool* (see Table 3). We ran the *Fisher's exact test* [35] that measures the statistical significance of categorical data (0/1 accuracy): *the average accuracy of submissions significantly differed by the choice of the tool for the steer tasks only.*

Question	Category	Time ($p < 0.05$)	Accuracy ($p < 0.05$)
*Q*1	Steer	**0.0007** (*)	**0.0033** (*)
*Q*2	Identify	$\mathbf{2.49 \times 10^{-5}}$ (*)	0.7475
*Q*3	Steer	**0.0043** (*)	**0.0202** (*)
*Q*4	compare (2)	**0.0154** (*)	1
*Q*5	compare (*N*)	$\mathbf{5.83 \times 10^{-6}}$ (*)	0.48
*Q*6	Customize	0.1207	0.0959

Table 3: Statistical significance of submission time and accuracy. (*) indicates statistical significance.

7.1.2 *Participants preferred NOAH to Excel.*

Figure 8 shows a diverging stacked bar chart representation of the survey results in which participants rated their experience with Excel and NOAH. For each metric mentioned in the study design section, there are two stacked bar charts, one for Excel and one for NOAH. Each component within a stacked bar represents the percentage of responses for the corresponding rating, where the ratings are on a scale of one one (strong disagreement) to seven (strong agreement). The average rating for each metric is represented with a white ellipse. Table 4 shows the results of the survey in which participants rated their experience with NOAH and Excel. Notably, NOAH had a higher average rating than Excel for all the metrics. In particular, participants felt that using NOAH was faster and easier compared to Excel. We further conducted a statistical significance test—the *Wilcoxon Signed-rank test* [97]—on the survey responses, that showed that for all the metrics, the ratings significantly differed by the choice of the tool, *i.e.*, NOAH or Excel.

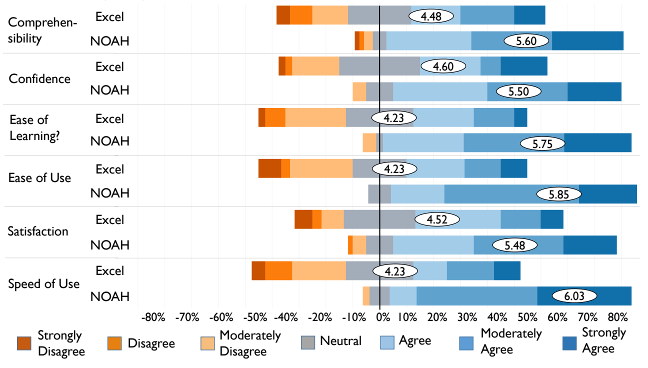


Figure 8: Participants found NOAH to be easier to use compared to Excel while being faster in completing tasks involving navigation.

Metric	NOAH	Excel	$p < 0.05$
Ease of Learning	μ = **5.75**, σ = 1.02	μ = 4.22, σ = 1.41	$\mathbf{1.49 \times 10^{-7}}$ (*)
Speed of Use	μ = **6.03**, σ = 0.99	μ = 4.22, σ = 1.65	$\mathbf{1.68 \times 10^{-7}}$ (*)
Ease of Use	μ = **5.88**, σ = 0.90	μ = 4.33, σ = 1.71	$\mathbf{7.85 \times 10^{-6}}$ (*)
Confidence	μ = **5.50**, σ = 1.79	μ = 4.60, σ = 1.50	**0.0096** (*)
Comprehensibility	μ = **5.60**, σ = 1.27	μ = 4.48, σ = 1.65	**0.0006** (*)
Satisfaction	μ = **5.48**, σ = 1.16	μ = 4.52, σ = 1.49	**0.0018** (*)

Table 4: Participants found NOAH to be easier to use compared to Excel while being faster in completing tasks involving navigation. (*) indicates statistical significance.

7.2 Exploration User Experience

To answer RQ2, we assess how NOAH's components, *i.e.*, the hierarchical overview, aggregate column, and context bar, impacted participants' navigation. For each observation, we present participant feedback from the interview phase.

7.2.1 *Dynamic Overview: Customizable Hierarchical Organization.*

Overall, the hierarchical overview prevented participants from being overwhelmed during exploration. Personalizing the overview enabled participants to define their own grouping of the data, resulting in a more meaningful overview presentation. However, the newer interactions at times deviated from spreadsheet semantics, contributing to a steeper learning curve.

Hierarchical overview aids exploration at scale. Participants found it difficult to perform various navigation tasks in Excel, especially on operations that required perusing data on multiple screens. NOAH, on the other hand, helped participants avoid endless scrolling via clicking and semantically zooming on the hierarchical overview, and provided cues for what to explore next via the bins of the overview. One participant ($P11$) commented—"*Excel can get overwhelming if you have a lot of data in it and sometimes with that data, finding things can be difficult*". Participants ($N = 6$) mentioned that they would prefer NOAH when the dataset is large: "*If I just had a large amount of data then I would prefer to use NOAH because then you would be able to see all of it [bins] at once*" ($P2$). NOAH's hierarchical overview helped participants comprehend the overall structure of the data better and prioritize the bin they want to visit next. One participant ($P5$) commented: "*I think it was just a little bit easier to navigate and find where things were because you could already see what bins had what.*" Another participant ($P1$) said: "*I like NOAH a lot better. It was a lot easier to look up different data and it was a lot quicker too*".

Overview customization enables related data to be analyzed together in task-specific ways. Bin customization enabled participants to personalize the overview based on their specific needs: "*I did like the fact that it lets you take a data sheet and, in some way, containerize the stuff you care and the stuff you don't care about (*$P16$*).*" Participants (14 out of 20) preferred the feature to Excel's filtering feature when working with numeric data—"*That was so much easier in NOAH than it was in Excel to be able to specify the range that you wanted it to go in*" ($P17$). Our analysis of the video recordings revealed that for the birdstrikes dataset in Excel, the customize task involved filtering out certain values from a total of 451 unique values. This manual filtering led to a significant delay in task completion, compared to the bin customization feature in NOAH. However, the time taken for this task was higher than other tasks in NOAH, as it required participants to restructure the overview before performing any calculation.

Overview customization interactions have a steeper learning curve. Unfamiliarity with the customization interactions in NOAH contributed to a higher completion time for the customize task compared to other tasks. The unfamiliarity led to some participants ($N = 5$) preferring Excel over NOAH for this task—"*Since I'm not used to spreadsheet data being presented that way, it took a little bit of getting used to*" ($P11$). Participants found some of the terminology used in the interface—*e.g.*, explore, bin—quite unfamiliar ($N = 14$). Moreover, two participants requested further clarification on how the bins were constructed.

Tradeoffs between hierarchical and flat overviews. While participants generally appreciated the hierarchical representation of the overview for numeric data, six participants stated that they would have preferred a flat overview for categorical data, where each bin corresponds to one item: "*I would prefer it start with all the bins split, and then I can merge them as I want (*$P13$*).*" Another participant ($P4$) said—"*When I started, it (NOAH) had already grouped them, I think, alphabetically. So, that creates an extra step in that I then have to go split them and then re-merge them.*"

7.2.2 *Aggregate Column: In-situ Steering-free Computation.*

The aggregate column feature enabled participants to avoid cumbersome steering interactions, resulting in faster and more accurate analysis compared to Excel. However, comparing the analysis results of multiple data subsets resulted in increased visual discontinuity.

Cumbersome steering replaced by a few button clicks with aggregate columns. The steer tasks required participants to issue a `COUNTIF` formula on a data subset. Participants found scrolling and steering in Excel to be cumbersome while issuing formulae—"*The one thing with Excel is I always try to go to the bottom of the data and type in the formula, and with something really long like this, the scrolling is a little bit cumbersome*" ($P4$). With NOAH, participants avoided steering by using the aggregate column feature on the menu-bar and selecting the appropriate formula. Multiple participants ($N = 13$) found it easier to issue formulae using this feature: "*There were some formulas to calculate that were definitely easier in NOAH because the aggregate column did all the work (*$P13$*).*" Another participant ($P3$) commented: "*And that creates convenience sort of because then you don't have to memorize anything and using the system becomes easier.*" However, two participants found the aggregation operations applied on the bins to be opaque compared to Excel where a user can directly manipulate the formula.

Issuing formulae is faster and more accurate with aggregate columns. While the accuracies and submission times for the steer tasks in Excel varied significantly across datasets, using NOAH, participants exhibited higher accuracies and faster submission times irrespective of the dataset (see Figure 6 and 7). The automated and steering-free aggregate column feature of NOAH contributed to high accuracies (100%) for the steer tasks: "*With NOAH, you don't have to highlight every number versus Excel where you actually have to select everything (*$P12$*).*" All of the 14 inaccurate submissions with Excel involved steering an incorrect spreadsheet region; 11 of the inaccurate submissions were with the Airbnb dataset. Analysis of screen recordings revealed that, for Excel, with the birdstrikes dataset, several participants used the *autosum* feature to quickly count the number of 1's in the binary-valued column involved in the steering task. Summing up binary values is equal to the number of 1's in the collection. Other participants used the status bar at the bottom of the spreadsheet that displayed the sum of the cells in the selected column. Participants avoiding data steering resulted in fewer errors. For the Airbnb dataset, participants had to manually steer the data as they could not use these shortcuts—the column involved in the steering task had 365 different values.

Visual discontinuity during comparisons while reduced, was not completely eliminated. For compare (N), participants had to perform N comparisons in NOAH while issuing the aggregate column once. However, the comparison among N bins resulted in increased visual discontinuity causing some ($N = 4$ out of 20) incorrect submissions. In Excel, the experience was worse, as the participants had to perform N separate steering tasks resulting in higher submission times of the the compare (N) tasks compared to compare (2) tasks (see Figure 6). In addition, the accuracies ($N = 13$ accurate) of the compare (N) task in Excel were lower.

7.2.3 *History, Context, and Coordination.*

The context bar enabled participants to revisit aggregation results of previously explored bins without having to reissue the aggregate column operation. The coordination between the overview and spreadsheet further helped participants relate the aggregate column results with the raw data.

History helps avoid repeated interactions. For the compare (2) task in NOAH, all of the participants used the context bar to navigate to a bin previously visited for the first steer task. As the bin currently being displayed was changed, the aggregate column was automatically updated to display values corresponding to that bin, enabling participants to view the aggregate column values instantly without having to reissue the operation. However, as Excel did not preserve any navigation history, participants had to re-execute the first steering operation. Therefore, the submission times for compare (2) tasks were faster in NOAH compared to Excel). One participant (*P*9) said—"*Noah was easy to find and compare and toggle in between (bins).*" One participant (*P*16) commented—"*Once I got familiar with the interface, it was easy to just say, I want to see this state, and I like that fact that like automatically it goes into the bins on NOAH.*"

Overview-spreadsheet coordination helps relate interactions on the overview with the raw data. The coordination between the hierarchical overview and spreadsheet in NOAH enabled users to quickly relate their interactions on the overview with the raw spreadsheet data. For example, for the find task, participants had to find all the cells within the spreadsheet that satisfied a condition corresponding to the preceding steer task. To do so, they had to either perform the conditional formatting operation to highlight relevant cells or skim through all the cells in the current window in Excel, resulting in higher completion times. In NOAH, participants benefited from having visual cues in the form of automatically colored cells, helping them relate the aggregate column with the raw data—"*You didn't have to do any additional steps and it was a visual cue right there, made it very quick to count it up (P17).*" Another participant (*P*9) commented—"*In Excel, you have to add your own condition for formatting. But you have to build that (conditional formatting) every time you need to ask a question. NOAH at least (has) something pre-built in, and you can easily count*" (*P*5).

7.3 User Study Limitations

Our study has a few limitations that can be strengthened by future larger-scale and finer-grained studies. Our participant pool demographics only partially represented the demographics of the general audience intended for NOAH. A larger sample with more participants with a range of skill-sets and backgrounds that better represents the spreadsheet user population would have provided more ecological validity to generalize our findings. Moreover, we only compared the performance of a NOAH-integrated spreadsheet with a traditional spreadsheet. We did not evaluate specific spreadsheet features like pivot table and SUBTOTAL. They violated many of the proposed design considerations; we discussed their limitations in the related work section. However, a future study targeted at evaluating the pros and cons of these features for spreadsheet exploration would be valuable. Finally, while we did present the impact of various components of NOAH, we did not isolate the effects of individual features during our study. For example, we did not study the effects of the binned overview (visual clarity versus visual continuity) and display layout (screen space trade-off) in isolation. Therefore, a more fine-grained study that teases apart the contribution of individual components of NOAH is warranted.

8 RELATED WORK

We discuss work that study interface limitations while exploring data, both inside and outside the context of spreadsheets.

Exploration in Popular Spreadsheet Systems. Spreadsheet systems support features that partially address the exploration challenges tackled by NOAH. Microsoft Excel enables users to manually create references to a spreadsheet region using the *named ranges* feature [82], accessible from the menu bar. Users can click on a named range to navigate to the referred region. However, the onus is on the user to create named ranges for each region of interest. The pivot table [96] and SUBTOTAL [63] features allow users to create a summary view to compare subsets of data without having to navigate to various locations within the sheet. The pivot table is placed in a separate disconnected region of the spreadsheet, preventing users from simultaneously accessing the data underlying the summary. The SUBTOTAL [63] function, on the other hand, embeds the summary with the raw data. However, for datasets with many subsets (*e.g.*, for numeric data), both the summary views can become very large, spanning multiple screens, and cause increased visual discontinuity during exploration. Google Sheets Explore [36] provides an overview of the data by auto-generating charts of data statistics. While Explore is a convenient means to understand high-level data characteristics, it doesn't address the challenges related to scrolling and steering.

Enhancing Spreadsheet Scalability. There has been some work on trying to make spreadsheets more scalable. Recent work has shown that users experience spreadsheets crashing and freezing even on millions of rows [59]. Some systems, *e.g.*, DataSpread [9], ABC [79], and Hillview [17], target trying to support large spreadsheets beyond main-memory limitations, by using scalable disk-based or distributed storage engines, or approximation. Other work has targeted making spreadsheets more interactive without necessarily increasing the size of the datasets they support [12, 78]. Our work is complementary—it instead targets users' perceptual challenges in exploration that befall even present-day spreadsheet systems that support fewer than a million rows; as our user study shows, participants frequently make mistakes when performing analytical tasks on systems like Excel. That said, NOAH is already integrated with DataSpread, and, while it has not been our present focus, NOAH can potentially be extended to support overviews on datasets that go beyond current spreadsheet limits using standard approximation approaches, such as that from Raman et al. [81], as well as online approximation approaches [41, 54].

Enhancing Spreadsheet Expressiveness. There has been a number of papers on trying to make spreadsheets more expressive, *e.g.*, for supporting joins [4, 5], SQL and relational algebra [13], and data entry and extraction via examples [6, 40]. Other work has targeted using a spreadsheet as a relational database engine [91], or extended databases to support spreadsheet-like primitives [98].

Tabular Data Summarization. There has been some limited work on summarizing tables. Smart drill-down [47] generates an interactive summary of a large table as a collection of rules; users can drill-down to a specific rule to view more fine-grained rules. Unlike

NOAH, users cannot use this summary to navigate to records of interest. Similar techniques have been applied to summarize machine learning [27, 32, 48] and aggregation results [94, 95].

Visual Interactive Spreadsheets. VisSh [72], SI [57], SSR [25], ASP [76], and PhotoSpread [50] extend the input/output capabilities of cells within spreadsheets, to display charts, animation, photos, or accept input via direct manipulation dialogs, among others. While these tools allow users to represent and manipulate data in a more flexible manner, they do not necessarily help users navigate data more effectively. Some recent work has extended spreadsheets to support navigation of semi-structured data [20].

Database Usability. There has been a lot of recent interest over the past decade in making databases and data exploration systems more usable [46]. A number of papers have targeted more intuitive and interactive specification interfaces for databases, including Dataplay [3], GestureDB [68], DBTouch [43], ETable [49], and QueryVis [56]. Other work has targeted query suggestion, wherein queries used by others are used to recommend or autocomplete user queries [18, 34, 52, 53, 64].

Understanding Spreadsheets and Spreadsheet Users. A number of papers have studied user behavior on spreadsheets. Some of these studies date back to the 90's [42, 69]; but even more recent studies find that users have cognitive burdens with spreadsheets, and that spreadsheets are rife with errors [55, 74, 102]. Some work has also studied and developed techniques for identifying higher-level structures within spreadsheets [44, 65].

Interactive Tables. TableLens [83] enables users to numerical information in tables, looking much like a spreadsheet with embedded bar charts. Cells out of focus display graphical bars proportional in length to the underlying values, providing a visual overview of the data, while cells within the user's current focus are magnified and display the graphical bars and the raw data. Similar ideas have been adopted by DataLens [8] for visualizing digital calendars, and by FOCUS [89] and InfoZoom [88] for exploring database query results. However, navigating (scrolling or steering) data that spans multiple screens is still cumbersome in TableLens.

Tabular Data Analysis (TDA) Tools. Visualization tools such as Tableau [90], Power BI [62], Keshif [101] and analytical tools such as SPSS [71], SAS [85], can all provide summaries of tabular data in various forms (visualizations, aggregate statistics). These summaries are static overviews of the data—much like pivot tables, these summaries are not dynamically linked to nor are co-located with the underlying raw data. For example, Keshif [101] can display all the unique values corresponding to an attribute of interest, *e.g.*, cities of the Airbnb data [29]. However, users cannot view or inspect the raw data corresponding to each city in a spreadsheet-like tabular setting, while being able to edit this raw data at will. With TDA tools, the spreadsheet look-and-feel is lost, and as a result, users lose the ability to directly manipulate raw data, derive new data, and issue formulae for free-form analysis.

Overall, none of the existing work provides solutions that enhance the exploration capabilities of users while simultaneously upholding the following goals of spreadsheet systems: a) facilitating the direct manipulation of data in-situ and b) enabling arbitrary derivation of new data and summaries using various exploration operations, *e.g.*, issuing formulae.

9 CONCLUSION

In this paper, we proposed a dynamic hierarchical overview to make spreadsheets more effective at supporting the exploration of large datasets that are increasingly the norm. We operationalized our proposed design in NOAH, a general-purpose spreadsheet navigation plug-in. NOAH employs an approximately equi-depth histogram construction strategy to compute a hierarchical overview on demand. Using the hierarchical overview, users were able to see a bird's eye view of the data, with the ability to scroll or seek additional details on demand, as well as employ dynamic aggregation in-situ, eliminating cumbersome steering operations. Quantitatively, we found that NOAH sped up navigation without compromising accuracy. Qualitatively, study participants identified NOAH as positively impacting their experience while overviewing, navigating, and performing computation on large datasets. Finally, we identified several enhancements to NOAH while discussing how the proposed overview plug-in can be integrated into existing spreadsheet tools.

REFERENCES

[1] 2020. DataSpread open-source software. https://github.com/dataspread/dataspread-web.

[2] 2020. Excel Specifications and Limits. https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3.

[3] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. 2012. DataPlay: interactive tweaking and example-driven correction of graphical database queries. In *UIST*. ACM, 207–218.

[4] Eirik Bakke, David Karger, and Rob Miller. 2011. A spreadsheet-based user interface for managing plural relationships in structured data. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2541–2550.

[5] Eirik Bakke and David R Karger. 2016. Expressive query construction through direct manipulation of nested relational results. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 1377–1392.

[6] Daniel W Barowy, Sumit Gulwani, Ted Hart, and Benjamin Zorn. 2015. FlashRelate: extracting relational data from semi-structured spreadsheets using examples. *ACM SIGPLAN Notices* 50, 6 (2015), 218–228.

[7] David V Beard and JOHN Q WALKER II. 1990. Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour & Information Technology* 9, 6 (1990), 451–466.

[8] Benjamin B Bederson, Aaron Clamage, Mary P Czerwinski, and George G Robertson. 2004. DateLens: A fisheye calendar interface for PDAs. *ACM Transactions on Computer-Human Interaction (TOCHI)* 11, 1 (2004), 90–119.

[9] Mangesh Bendre et al. 2018. Towards a Holistic Integration of Spreadsheets with Databases: A Scalable Storage Engine for Presentational Data Management. In *ICDE*.

[10] Mangesh Bendre, Bofan Sun, Ding Zhang, Xinyan Zhou, Kevin Chen-Chuan Chang, and Aditya Parameswaran. 2015. DATASPREAD: Unifying Databases and Spreadsheets. *Proceedings of the VLDB Endowment* 8, 12 (2015).

[11] Mangesh Bendre, Bofan Sun, Ding Zhang, Xinyan Zhou, Kevin Chen-Chuan Chang, and Aditya Parameswaran. 2015. DataSpread: Unifying Databases and Spreadsheets. *PVLDB* 8, 12 (2015), 2000–2003. https://doi.org/10.14778/2824032.2824121

[12] Mangesh Bendre, Tana Wattanawaroon, Kelly Mack, Kevin Chang, and Aditya Parameswaran. 2019. Anti-freeze for large and complex spreadsheets: Asynchronous formula computation. In *Proceedings of the 2019 International Conference on Management of Data*. 1277–1294.

[13] Mangesh Bendre, Tana Wattanawaroon, Sajjadur Rahman, Kelly Mack, Yuyang Liu, Shichu Zhu, Yu Lu, Ping-Jing Yang, Xinyan Zhou, Kevin Chen-Chuan Chang, Karrie Karahalios, and Aditya G. Parameswaran. 2019. Faster, Higher, Stronger: Redesigning Spreadsheets for Scale. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. 1972–1975.

[14] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.

[15] David A Bradbard, Charles Alvis, and Richard Morris. 2014. Spreadsheet usage by management accountants: An exploratory study. *Journal of Accounting Education* 32, 4 (2014), 24–30.

[16] Matthew Brehmer and Tamara Munzner. 2013. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2376–2385.

[17] Mihai Budiu, Parikshit Gopalan, Lalith Suresh, Udi Wieder, Han Kruiger, and Marcos K Aguilera. 2019. Hillview: A trillion-cell spreadsheet for big data.

Proceedings of the VLDB Endowment 12, 11 (2019), 1442–1457.
[18] Ugur Cetintemel et al. 2013. Query Steering for Interactive Data Exploration.. In *CIDR*.
[19] Yolande E Chan and Veda C Storey. 1996. The use of spreadsheets in organizations: Determinants and consequences. *Information & Management* 31, 3 (1996), 119–134.
[20] Kerry Shih-Ping Chang and Brad A Myers. 2016. Using and exploring hierarchical data in spreadsheets. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2497–2507.
[21] Zhe Chen and Michael Cafarella. 2013. Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search over the Web*. 1–8.
[22] Zhe Chen and Michael Cafarella. 2014. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1126–1135.
[23] Zhe Chen, Michael Cafarella, Jun Chen, Daniel Prevo, and Junfeng Zhuang. 2013. Senbazuru: A prototype spreadsheet database management system. *VLDB* 6, 12 (2013), 1202–1205.
[24] Zhe Chen, Sasha Dadiomov, Richard Wesley, Gang Xiao, Daniel Cory, Michael Cafarella, and Jock Mackinlay. 2017. Spreadsheet property detection with rule-assisted active learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 999–1008.
[25] EH-H Chi, Phillip Barry, John Riedl, and Joseph Konstan. 1997. A spreadsheet approach to information visualization. In *Information Visualization, 1997. Proceedings., IEEE Symposium on*. IEEE, 17–24.
[26] Fernando Chirigati, Harish Doraiswamy, Theodoros Damoulas, and Juliana Freire. 2016. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 1011–1025.
[27] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2019. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1550–1553.
[28] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. 2009. A review of overview+ detail, zooming, and focus+ context interfaces. *ACM Computing Surveys (CSUR)* 41, 1 (2009), 2.
[29] Murray Cox. 2020. The inside airbnb dataset. Retrieved May 5, 2020 from http://insideairbnb.com/get-the-data.html
[30] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2015. Vizdom: interactive analytics through pen and touch. *PVLDB* 8, 12 (2015), 2024–2027.
[31] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. TableSense: Spreadsheet Table Detection with Convolutional Neural Networks. In *Thirty-Third AAAI Conference on Artificial Intelligence*.
[32] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and informative explanations of outcomes. *Proceedings of the VLDB Endowment* 8, 1 (2014), 61–72.
[33] Niklas Elmqvist and Jean-Daniel Fekete. 2009. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (2009), 439–454.
[34] Ju Fan, Guoliang Li, and Lizhu Zhou. 2011. Interactive SQL query suggestion: Making databases user-friendly. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 351–362.
[35] Ronald A Fisher. 1922. On the interpretation of χ 2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94.
[36] Google. 2020. Google Explore. Retrieved May 5, 2020 from https://support.google.com/docs/answer/6280499
[37] Google. 2020. Google Sheets scale. Retrieved May 5, 2020 from https://developers.google.com/drive/answer/37603
[38] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery* 1, 1 (1997), 29–53.
[39] Jonathan Grudin. 2001. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 458–465.
[40] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* 46, 1 (2011), 317–330.
[41] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. 1997. Online aggregation. In *Acm Sigmod Record*, Vol. 26. ACM, 171–182.
[42] David G Hendry and Thomas RG Green. 1994. Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model. *International Journal of Human-Computer Studies* 40, 6 (1994), 1033–1065.
[43] Stratos Idreos and Erietta Liarou. 2013. dbTouch: Analytics at your Fingertips.. In *CIDR*.
[44] Takeo Igarashi, Jock Mackinlay, Bay-Wei Chang, and Polle T. Zellweger. 1998. Fluid visualization of spreadsheet structures. In *Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on*. IEEE, 118–125. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=706154
[45] Yannis Ioannidis. 2003. The history of histograms (abridged). In *Proceedings 2003 VLDB Conference*. Elsevier, 19–30.
[46] H. V. Jagadish et al. 2007. Making database systems usable. In *SIGMOD*. ACM, 13–24. http://dl.acm.org/citation.cfm?id=1247483
[47] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2015. Smart drill-down: A new data exploration operator. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1928–1931.
[48] Minsuk Kahng, Dezhi Fang, and Duen Horng Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 1–6.
[49] Minsuk Kahng, Shamkant B Navathe, John T Stasko, and Duen Horng Polo Chau. 2016. Interactive Browsing and Navigation in Relational Databases. *Proceedings of the VLDB Endowment* 9, 12 (2016).
[50] Sean Kandel, Andreas Paepcke, Martin Theobald, Hector Garcia-Molina, and Eric Abelson. 2008. Photospread: a spreadsheet for managing photos. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1749–1758.
[51] Miranda Katz. 2017. A Lone Data Whiz Is Fighting Airbnb—and Winning. Retrieved May 5, 2020 from https://www.wired.com/2017/02/a-lone-data-whiz-is-fighting-airbnb-and-winning/
[52] Nodira Khoussainova et al. 2009. A Case for A Collaborative Query Management System.. In *CIDR*.
[53] Nodira Khoussainova et al. 2010. SnipSuggest: Context-aware autocompletion for SQL. *VLDB Endowment* 4, 1 (2010), 22–33. http://dl.acm.org/citation.cfm?id=1880175
[54] A Kim, L Xu, T Siddiqui, S Huang, S Madden, and A Parameswaran. 2016. *Optimally Leveraging Density and Locality to Support LIMIT Queries*. Technical Report. Available at: https://arxiv.org/pdf/1611.04705.pdf.
[55] Barry R Lawson, Kenneth R Baker, Stephen G Powell, and Lynn Foster-Johnson. 2009. A comparison of spreadsheet users with different levels of experience. *Omega* 37, 3 (2009), 579–590.
[56] Aristotelis Leventidis, Jiahui Zhang, Cody Dunne, Wolfgang Gatterbauer, HV Jagadish, and Mirek Riedewald. 2020. QueryVis: Logic-based diagrams help users understand complicated SQL queries faster. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2303–2318.
[57] Marc Levoy. 1994. Spreadsheets for images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 139–146.
[58] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. imMens: Real-time Visual Querying of Big Data. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 421–430.
[59] Kelly Mack, John Lee, Kevin Chen-Chuan Chang, Karrie Karahalios, and Aditya G. Parameswaran. 2018. Characterizing Scalability Issues in Spreadsheet Software using Online Forums. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*.
[60] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.
[61] Microsoft. 2020. How finance leaders can drive performance. Retrieved May 5, 2020 from https://enterprise.microsoft.com/en-gb/articles/roles/finance-leader/how-finance-leaders-can-drive-performance/
[62] Microsoft. 2020. Power BI. Retrieved May 5, 2020 from https://powerbi.microsoft.com/en-us/
[63] Microsoft. 2020. SUBTOTAL. Retrieved May 5, 2020 from https://support.office.com/en-us/article/subtotal-function-7b027003-f060-4ade-9040-e478765b9939
[64] Tova Milo and Amit Somech. 2018. Next-step suggestions for modern interactive data analysis platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 576–585.
[65] Roland Mittermeir and Markus Clermont. 2002. Finding high-level structures in spreadsheet programs. In *Reverse Engineering, 2002. Proceedings. Ninth Working Conference on*. IEEE, 221–232. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1173080
[66] Dominik Moritz, Bill Howe, and Jeffrey Heer. 2019. Falcon: Balancing interactive latency and resolution sensitivity for scalable linked visualizations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–11.
[67] Hamid Mousavi and Carlo Zaniolo. 2011. Fast and accurate computation of equi-depth histograms over data streams. In *Proceedings of the 14th International Conference on Extending Database Technology*. ACM, 69–80.
[68] Arnab Nandi, Lilong Jiang, and Michael Mandel. 2013. Gestural Query Specification. *VLDB Endowment* 7, 4 (2013). http://web.cse.ohio-state.edu/~jianglil/files/gestureQuerySpecification.pdf
[69] Bonnie A Nardi and James R Miller. 1990. *The spreadsheet interface: A basis for end user programming*. Hewlett-Packard Laboratories.
[70] Chris North and Ben Shneiderman. 2000. Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies* 53, 5 (2000), 715–739.
[71] Marija J Norušis. 1986. *The SPSS guide to data analysis*. Spss.
[72] Fabian Nunez and Edwin Blake. 2000. Vissh: A data visualisation spreadsheet. In *Data Visualization 2000*. Springer, 209–218.

[73] Bureau of Transportation Statistics. 2020. The US flight dataset. Retrieved May 5, 2020 from https://www.transtats.bts.gov/

[74] Raymond R Panko. 1998. What we know about spreadsheet errors. *Journal of Organizational and End User Computing (JOEUC)* 10, 2 (1998), 15–21.

[75] Ken Perlin and David Fox. 1993. Pad: an alternative approach to the computer interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, 57–64.

[76] Kurt W Piersol. 1986. Object-oriented spreadsheets: the analytic spreadsheet package. In *ACM Sigplan Notices*, Vol. 21. ACM, 385–390.

[77] Neil Raden. 2005. Shedding light on shadow IT: Is Excel running your business. *DSSResources.com* 26 (2005).

[78] Sajjadur Rahman, Kelly Mack, Mangesh Bendre, Ruilin Zhang, Karrie Karahalios, and Aditya Parameswaran. 2020. Benchmarking Spreadsheet Systems. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1589–1599.

[79] Vijayshankar Raman et al. 1999. Scalable Spreadsheets for Interactive Data Analysis. In *ACM SIGMOD Workshop on DMKD*.

[80] Vijayshankar Raman, Bhaskaran Raman, and Joseph M Hellerstein. 1999. Online dynamic reordering for interactive data processing. In *VLDB*, Vol. 99. 709–720.

[81] Vijayshankar Raman, Bhaskaran Raman, and Joseph M Hellerstein. 2000. Online dynamic reordering. *The VLDB Journal* 9, 3 (2000), 247–260.

[82] Named ranges. 2020. Define and use names in formulas. Retrieved May 5, 2020 from https://www.excel-easy.com/examples/names-in-formulas.html

[83] Ramana Rao and Stuart K Card. 1994. 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*. ACM, 318–322.

[84] Jonathan C Roberts. 2007. State of the art: Coordinated & multiple views in exploratory visualization. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*. IEEE, 61–71.

[85] Institute SAS. 1985. SAS user's guide: statistics. *SAS Institute, Cary, NC* (1985).

[86] Ben Shneiderman. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*. Elsevier, 364–371.

[87] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *PVLDB* 10, 4 (2016), 457–468.

[88] Michael Spenke and Christian Beilken. 2003. Visualization and Analysis of Formula One Racing Results with InfoZoom-the Demo.. In *INTERACT*.

[89] Michael Spenke, Christian Beilken, and Thomas Berlage. 1996. FOCUS: the interactive table for product comparison and selection. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*. ACM, 41–50.

[90] Tableau. 2020. Tableau Software. Retrieved May 5, 2020 from https://www.tableau.com/

[91] Jerzy Tyszkiewicz. 2010. Spreadsheet as a relational database engine. In *SIGMOD*. ACM, 195–206.

[92] Michelle Q Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. 2000. Guidelines for using multiple views in information visualization. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 110–119.

[93] Jennifer Watts-Perotti and David D Woods. 1999. How experienced users avoid getting lost in large display networks. *International Journal of Human-Computer Interaction* 11, 4 (1999), 269–299.

[94] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. 2018. Interactive summarization and exploration of top aggregate query answers. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. 2196.

[95] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. 2018. Qagview: Interactively summarizing high-valued aggregate query answers. In *Proceedings of the 2018 International Conference on Management of Data*. 1709–1712.

[96] Wikipedia. 2020. Pivot Table. Retrieved May 5, 2020 from https://en.wikipedia.org/wiki/Pivot_table

[97] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[98] Andrew Witkowski, Srikanth Bellamkonda, Tolga Bozkaya, Gregory Dorman, Nathan Folkert, Abhinav Gupta, Lei Shen, and Sankar Subramanian. 2003. Spreadsheets in RDBMS for OLAP. In *SIGMOD*. ACM, 52–63. http://dl.acm.org/citation.cfm?id=872767

[99] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization & Computer Graphics* 1 (2016), 1–1.

[100] Eugene Wu, Leilani Battle, and Samuel R Madden. 2014. The case for data visualization management systems: vision paper. *Proceedings of the VLDB Endowment* 7, 10 (2014), 903–906.

[101] Mehmet Adil Yalçın, Niklas Elmqvist, and Benjamin B Bederson. 2018. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE transactions on visualization and computer graphics* 24, 8 (2018), 2339–2352.

[102] Pingjing Yang, Ti-Chung Cheng, Sajjadur Rahman, Mangesh Bendre, Karrie Karahalios, and Aditya Parameswaran. 2020. Understanding Data Analysis Workflows on Spreadsheets: Roadblocks and Opportunities. In *Proceedings of Workshop on Human-In-the-Loop Data Analytics (HILDA'20)*.