NOAH: Interactive Spreadsheet Exploration with Dynamic Hierarchical Overviews

Sajjadur Rahman*
Megagon Labs
sajjadur@megagon.ai

Mangesh Bendre*
VISA Research
mbendre@visa.com

Yuyang Liu
U Illinois (UIUC)
yuyangl2@illinois.edu

Shichu Zhu*
Google LLC
shichuzhu@google.com

Zhaoyuan Su*
UC Irvine
nick.su@uci.edu

Karrie Karahalios
U Illinois (UIUC)
kkarahal@illinois.edu

Aditya G. Parameswaran*
UC Berkeley
adityagp@berkeley.edu

ABSTRACT

Spreadsheet systems are by far the most popular platform for data exploration on the planet, supporting millions of rows of data. 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 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 overview to explore the data at various granularities, zooming in and out of the spreadsheet. They can issue formulae over data subsets without cumbersome scrolling or range selection, enabling users to gain a high or low-level perspective of the spreadsheet. An implementation of our dynamic hierarchical overview, NOAH, integrated within DataSpread, preserves spreadsheet semantics and look and feel, while introducing such enhancements. Our user studies demonstrate that NOAH makes it more intuitive, easier, and faster to navigate spreadsheet data compared to traditional spreadsheets like Microsoft Excel and spreadsheet plug-ins like Pivot Table, for a variety of exploration tasks; participants made fewer mistakes in NOAH while being faster in completing the tasks.

PVLDB Reference Format:
Sajjadur Rahman, Mangesh Bendre, Yuyang Liu, Shichu Zhu, Zhaoyuan Su, Karrie Karahalios, and Aditya G. Parameswaran. NOAH: Interactive Spreadsheet Exploration with Dynamic Hierarchical Overviews. PVLDB, 14(6): 970 - 983, 2021.
doi:10.14778/3447689.3447701

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/dataspread/NOAH.

*This work began when these authors were part of the University of Illinois.

Proceedings of the VLDB Endowment, Vol. 14, No. 6 ISSN 2150-8097.
doi:10.14778/3447689.3447701

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 [49]. Data analysts prefer to operate on data within spreadsheets while shunning BI tools with more advanced analytical features [13, 65]. 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 [57].

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 [26], a 12.5× increase from the previous limit of 400K cells, while Microsoft Excel now supports a million rows [48]. 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 [80]. *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.*

While exploring spreadsheet datasets that span multiple screens, ***users often lose context [76], become overwhelmed [28], and introduce errors [61]***, as detailed below:

1. 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 context available is the primitive scrollbar indicating the user's location; users are forced to remember the layout and structure of the spreadsheet during navigation [76].

2. Visual discontinuities. Users can view a screenful of data at a time, introducing visual discontinuities between data being displayed. Comparing spatially separated subsets of data requires moving back and forth across screens. As an alternative, users tend to copy subsets of data side-by-side [57], which is cumbersome and error-prone.

3. Cognitive and mechanical burdens. To avoid getting lost, users often end up taking drastic measures e.g., by sketching maps of spreadsheets on paper, or adding landmarks to the spreadsheet in color [76]. Issuing a formula on a data range spanning multiple screens is problematic: users may use *steering*, i.e., they click and

drag the mouse to select the data, or enter the data range manually having memorized it. Both approaches lead to errors due to incorrect range selection [61].

Despite spreadsheets being around for nearly five decades, little work has addressed the aforementioned challenges that stem from navigating and exploring datasets spanning several screens [80]. Existing features such as Pivot Tables, named ranges, and subtotals, partially alleviate some 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 ranges of data, a replacement for sketches made on paper [76]. Other work tries to make spreadsheets more scalable [5, 68], responsive [6, 69], and expressive [2, 3, 7, 74], but none of these papers address the usability challenges underlying spreadsheet data exploration that occur on even modest-sized datasets. We discuss other related work in Section 8.

To support effective exploration on spreadsheets, one approach is to integrate an overview of the overall structure of the data alongside the spreadsheet [28], giving an *overview+detail* interface [19]. Overview+detail interfaces have been applied to text editors and maps [19] to reduce cognitive load for users during navigation by providing them with the big picture first, along with zooming in and out on demand. Unfortunately, in the spreadsheet context, *simply providing a zoomed out view as an overview is unreadable and therefore unusable*. The overview must provide a customizable summary of the spreadsheet, while allowing users to connect the summary to the raw spreadsheet data, with the following challenges:

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 as users navigate the spreadsheet? How do we ensure that this overview *facilitates the search* for individual rows or groups of rows of interest? 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 on zooming in and out? How do we allow the users to *customize the grouping*, if the automated grouping is not semantically meaningful?

Overview capabilities and integration. The overview must be a plug-in that enhances the capabilities of spreadsheets, as opposed to a potentially jarring or confusing replacement for spreadsheets. How can we *seamlessly integrate* our overview with current spreadsheet tools without impacting semantics or look-and-feel? How do we *facilitate simple interactions* on the overview that help users avoid endless scrolling during exploration and error-prone steering while issuing formulae across multiple screenfuls of data? As the user performs 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 is 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 [5], an open-source spreadsheet. While NOAH's design is not tied to DataSpread, we opted not to use other 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.

Contributions. The primary contributions of our work are:

- We introduce the notion of *dynamic hierarchical overviews* to eliminate cumbersome scrolling and steering during exploration of 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 conduct two user studies, based on several spreadsheet exploration tasks, to evaluate the benefits and limitations of this plugin. The first study explores NOAH impacts spreadsheet exploration performance; 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. The second study compares participants' spreadsheet exploration experience with Pivot Table and NOAH; NOAH improved overall task accuracy by 20.78% while reducing the overall task completion time by 28.47% relative to Pivot Table.

2 DESIGN CRITERIA AND USE CASES

We now present our design considerations for developing NOAH. We then discuss a use-case that captures how NOAH addresses the challenges associated with different types of exploration tasks.

2.1 Design Considerations

Our design considerations were informed by prior work in visualization [72], overview+detail interfaces [19] and multiple-coordinated views [75].

DC1. Construct the overview in-situ. Maintaining the overview in a separate location from the spreadsheet 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 [28].

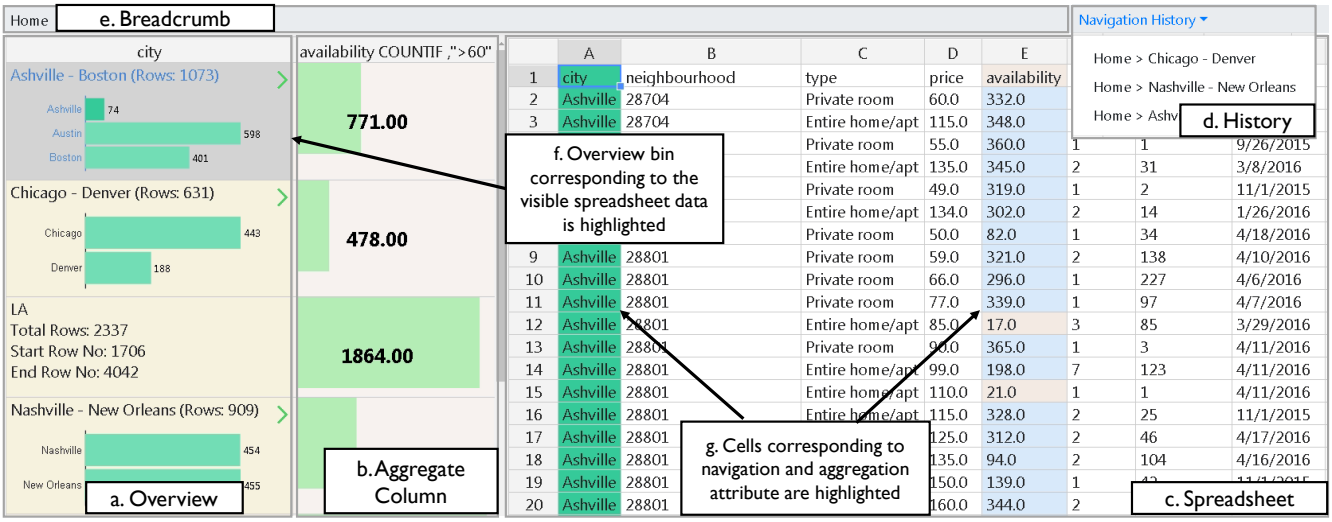


Figure 1: The NOAH interface consisting of (a) a zoomable overview and (b) an aggregate column integrated with (c) a spreadsheet. A context bar consisting of (d) a navigation history displaying locations visited so far using the overview, and (e) a breadcrumb showing the current navigation path (*e.g.*, 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.

DC2. Ensure reduced visual discontinuity while providing details-on-demand. To enhance visual continuity while issuing formulae, instead of cumbersome steering, the interface should enable interactions on the overview to perform such computations on demand and view results [72]. The interface should maintain visual continuity as users navigate to different spreadsheet locations, automatically recomputing formulae to reflect the new focus.

DC3. Balance the overview screen space. The limited screen real-estate leads to a trade-off between visual discontinuity (*DC2*) and clarity. A fine-grained overview improves visual clarity while increasing discontinuity: users need to scroll through the overview to access distant subsets of data. A coarse-grained overview decreases discontinuity at the cost of clarity—the overview may span too many data subsets and appear cluttered.

DC4. Enable overview-spreadsheet coordination. Since users view the overview and spreadsheet simultaneously, interactions on one should be *linked* or reflected on the other [71, 75].

DC5. Facilitate customization of the overview. As the overview is automatically generated, it may not reflect domain-specific context [68]. For example, an overview constructed on a grading spreadsheet by binning nearby scores may not match the instructor-intended letter grade ranges. User-driven customization is essential.

DC6. Display navigation history. The interface should record navigation history, allowing users to revisit previous locations [72], while also displaying their current navigation path for context.

2.2 NOAH: Tasks and Use-cases

Table 1 captures the types of spreadsheet tasks for which NOAH enhances user experience, drawn from an exhaustive list by Brehmer et al. [10]. This list encompasses a range of domain-independent tasks on visual data representations, developed after analyzing task classification systems in over two dozen papers, and has been applied to several scenarios, such as developing models for visualization systems [47], designing task taxonomies for cartograms [45], and defining the scope of tasks in various domains, *e.g.*, interactive task authoring [24], document mining [9], multivariate network analysis [64], and mass cytometry [30].

Table 1: Use cases where NOAH enhances exploration experience.

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

Usage Scenario. Amy, a journalist, is exploring the *Inside Airbnb* dataset [20], a spreadsheet dataset of all the Airbnb listings across US cities with $\approx 100k$ records and 15 attributes. This dataset was created to investigate whether certain 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 [36]. Given that this is the first time she's examining this dataset, Amy wants to first gain a bird's eye view of the data. Without NOAH, Amy 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 Amy 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. With NOAH, she organizes by city—with NOAH providing an automatically constructed high-level *hierarchical* overview of cities (Figure 1a). She then uses this overview to start exploration, understanding which cities are present, and roughly how many listings does each city have. 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, essentially zooming into the

overview via an an operation akin to OLAP drill-down [27]. For example, as shown in Figure 1c, clicking on the *Ashville-Boston* bin displays the Ashville listings (`locate`).

Next, say Amy 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, Amy 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), much like aggregation results in cross tabs [27], or aggregates in `GROUP BY` queries. 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 ensures coordination and 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 Amy 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.

3 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.

3.1 Overview Design

Spreadsheets often have one or more tabular regions containing structured data [5], 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, our overview of the Airbnb data on the navigation attribute "city" has four bins at the highest (coarsest) granularity level. 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 also supports other interactions *e.g.*, customization and aggregation, discussed later.

3.1.1 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 (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 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.* We show various examples of coupled and decoupled interactions in Section 3.3, *e.g.*, clicking, semantic zooming, and aggregate column creation. We explain the implementation details of NOAH-DataSpread coordination in Section 4.

3.1.2 The Rationale for Binning. Overviews within popular interfaces are often designed as a spatially partitioned collection of thumbnails on the left of the standard detailed view, similar to Microsoft Power Point or Adobe Reader. To strike a balance between the objectives of visual discontinuity and clarity (**DC3**), we instead designed an overview that abstracts the data at varying levels of detail, in a hierarchical or multi-granularity fashion. Presenting information at multiple granularities makes visual representations more perceptually scalable and less cluttered [22]. 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 [19]. Users can resize the overview to control the 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* [42]. 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.

3.2 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 3.5. Since current spreadsheet systems only support a few million rows, the overview construction is performed in-memory and is extremely fast.

3.2.1 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 3.5. When the user requests 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 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. 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 3.2.3. 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 partitions 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 [32], distribution fitting in data streams [55], and interactive scrolling on large spreadsheets [69]. 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 $b_j = \frac{n}{k}, \forall j \in [1, k]$.*

If 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.

3.2.2 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 $i \in [0, m]$, $j \in [0, k]$

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

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. For example, for $H(2, 2)$ the possible split points are $d \in \{1, 2\}$. The utility of $H(1, 1)$ (when $d = 1$) and $H(0, 1)$ (when $d = 2$) are already known. We choose the split point that yields a new bin—constructed from the remaining unique values (c_l)—whose size is closest to the expected equi-depth bin size ($\hat{b}$). The complexity of this algorithm is $O(m^2 k)$.

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 $v_i \leq u_{j-1}$ 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.

3.2.3 Extending to a Hierarchy. Our dynamic hierarchical overview is constructed on-demand top-down (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 bins are displayed. If the user drills down or zooms into into any of these k bins (using actions that we will discuss in Section 3.3), 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. 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.

Note that our hierarchical overview 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

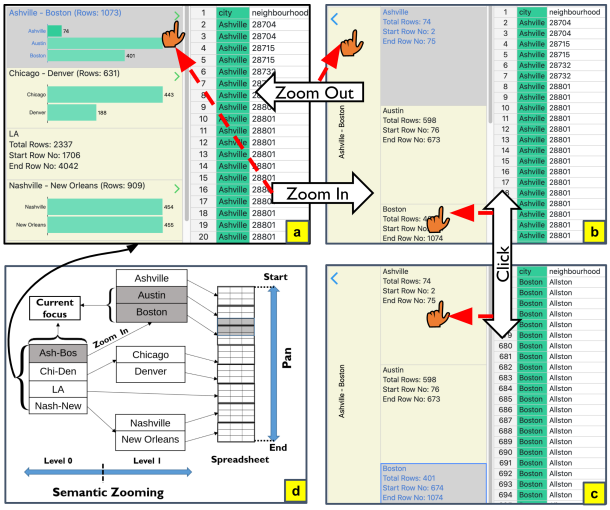


Figure 2: Navigational operations. (a) The overview at the highest level of granularity. (b) A zoomed in view of the *Ashville-Boston* bin. (c) As the user clicks on the *Boston* bin, the Boston listings are displayed on the sheet. The *Boston* bin is highlighted in gray to indicate user's current focus.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

fewer than k children. Moreover, to guarantee efficient retrieval and maintenance, B+-trees impose a constraint on the number of keys per internal node which cannot be violated; we have no such restriction. Furthermore, 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 was to ensure a near-equal subdivision of records, making it easier for human consumption, instead of efficient access.

3.3 Operations and Interactions

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

3.3.1 Clicking and Semantic Zooming. Clicking a bin is an example of an interaction that is coupled between the spreadsheet and the overview (see Section 3.1.1). 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).

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* [62]. 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 3.2.2, 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.

3.3.2 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. 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 numeric, users can split the bin into an arbitrary number of bins.

3.4 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 to issue formulae—with the formulae having been declared once as part of the overview, and dynamically updated as users traverse 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.

3.4.1 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 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. 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. 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 [27]; here, we are drilling down/rolling up on automatically grouped values of a given attribute rather than across multiple attributes.

3.4.2 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 *value bar*, whose length is proportional

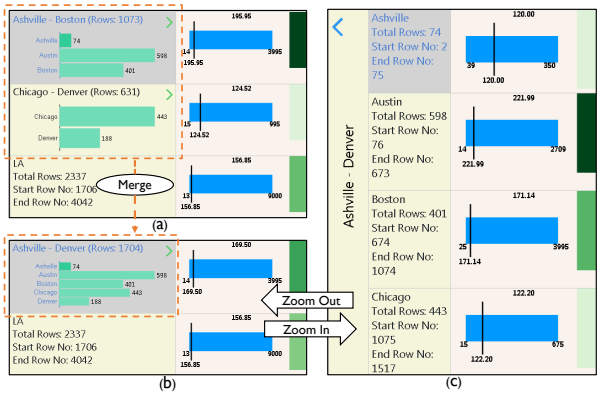


Figure 3: (a) Chart view of the aggregate column. (b) New bin created by merging the top two bins. (c) Zooming into the new bin.

to the corresponding aggregate (see Figure 1b), permitting visual comparison across bins. The chart representation varies depending on the formula category: 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. The chart representations are designed based on existing best practices [44, 78], which are encoded as rules in NOAH. We discuss the design choices in detail in the technical report [66].

3.4.3 Constructing the Aggregate Column. The aggregate column is kept in sync with the bins as users zoom in and out. 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 use the context bar to explore previously visited overview bins (**DC6**), comprising 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 sorted data, which is fairly efficient.

3.5 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 practice, 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 [4, 14–17, 21] 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. 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 [5], 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 (see our technical report [66] for details.) 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.

Multiple Navigation Attributes. Our hierarchy can be constructed for multiple attributes (*e.g.*, explore the Airbnb data by city and neighborhood). One approach 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. This is similar to how indexes on combinations of attributes, *e.g.*, B+-tree, are organized.

Supporting Dynamic Crossfiltering. Currently, the charts displayed in an *aggregate column* are non-interactive (see Figure 3), *i.e.*, users cannot interact with the charts to visually filter relevant or interesting data points within the spreadsheet. Such linked selection is similar to crossfilter [54, 79] where one action may generate hundreds of filter operations per second. As discussed in Section 3.1, each bin in the 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.

4 SYSTEM ARCHITECTURE

We have integrated NOAH with DataSpread [5], a web-based spreadsheet, as a plug-in (see Figure 4). The NOAH client is responsible for capturing user interactions on the overview (see Section 3.3), and for rendering the overview. Given a user interaction, the NOAH *request processor* issues a request to the back-end NOAH *controller* which propagates the request to the appropriate manager. The *hierarchy* and *aggregation* managers are responsible for the creation and maintenance of the hierarchy and aggregate column(s), respectively. The NOAH back-end materializes the hierarchy and

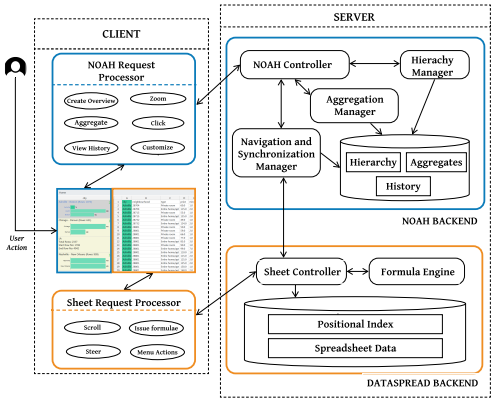


Figure 4: System architecture.

aggregation results. The *navigation and synchronization manager* coordinates with the DataSpread *sheet controller* to update the spreadsheet view based on the user's interactions 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.

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 [5], 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 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 3.2.2) on the attribute values. As each attribute value is associated with a *row id*, the bins 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. NOAH can also be potentially integrated with systems like Excel and Google Sheets, since these systems all support access to cells by position, as is needed for formula computation. We discuss this integration further in our technical report [66].

5 STUDY 1: NOAH VS. SPREADSHEET TOOLS

We now 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. Our study was designed to explore the following questions derived from our design considerations in Section 2.1: a) how does a dynamic hierarchical overview like NOAH impact the efficiency of spreadsheet data exploration? and b) how do the various components of NOAH impact users' spreadsheet exploration experiences?

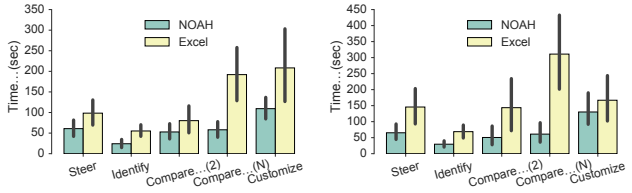
Study Design and Participants. The two systems used in the study were: Microsoft Excel, and NOAH integrated as a plug-in within DataSpread [5] (henceforth, referred to as NOAH). We recruited 20 participants (11 female, 9 male) via a university email newsletter. To ensure that prior experience with spreadsheets 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, and (c) a semi-structured interview and surveys to collect qualitative feedback regarding the quiz phase.

Datasets. We used two datasets—birdstrikes [79] and airbnb [20]. 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 has a similar number of attributes—15 attributes; six categorical, two spatial, one temporal, and six numeric). However, the airbnb dataset is larger than the birdstrikes dataset. As we alternated the datasets and systems between consecutive participants, we wanted to maintain consistency of datasets. So, to ensure parity and that the dataset size would not impact the performance with a specific system, we created a sample of the original airbnb dataset with 10,925 records, by uniformly sampling 10% of the records from each city.

Table 2: Quiz tasks for airbnb dataset. The task purposes and use cases correspond to Table 1. DC = design consideration (in Section 2).

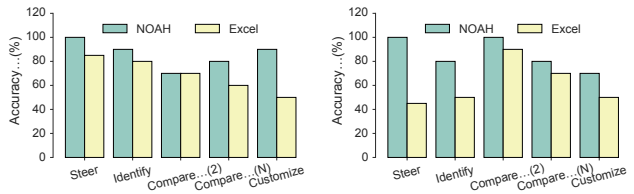
Category	Question (Q), Purpose (P), Use case (U)
steer	**Q**: Organize the data by City. How many listings in Chicago were available more than 60 days (*availability* > 60)? [**DC2**, **DC3**], **P**: *Search*→*Analyze*, **U**: `lookup`→`identify`
find	**Q**: How many listings in the currently visible spreadsheet window have availability > 60? [**DC1**, **DC4**], **P**: *Search*, **U**: `browse`
steer	**Q**: How many listings in Santa Cruz were available more than 60 days (*availability* > 60)?, [**DC2**, **DC3**] **P**: *Search*→*Analyze*, **U**: `lookup`→`identify`
compare (2)	**Q**: Which of the following two cities has a higher number of listings with availability > 60: (a) Chicago, (b) Santa Cruz? [**DC2**, **DC6**], **P**: *Analyze*, **U**: `compare`
compare (N)	**Q**: Find the city with the most listings with availability > 60. [**DC2**], **P**: *Analyze*→*Search*, **U**: `summarize`→`locate`
customize	**Q**: Organize the data by *availability*. How many listings that are rented as "Entire Home" are available between 46 to 157 days? [**DC5**], **P**: *Analyze*→*Search*→*Produce*, **U**: `generate`→`summarize`→`lookup`

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*, *Analyze*, 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 [8, 39]. The tasks were presented in the same order as shown in Table 2 for



(a) birdstrikes (b) airbnb

Figure 5: Submission times per category. Submission times are much smaller for NOAH on average compared to Excel.



(a) birdstrikes (b) airbnb

Figure 6: Per category accuracy indicates higher accuracy of task completion in NOAH compared to Excel.

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 these measures with 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 systems. We then measured the statistical significance of the comparisons between the two systems.

Limitations. The participant pool of the study only partially represented the demographics of the general spreadsheet audience. A larger participant pool with a range of skill-sets and backgrounds that better represents the spreadsheet user population would have provided more generality. Moreover, we compared NOAH integrated within DataSpread with Excel instead of DataSpread. Since DataSpread lacks many functionalities available in systems like Excel, we opted to compare against Excel. While using two different spreadsheet systems is a potential limitation, we combined our quantitative analysis with qualitative survey, interview, and screen/audio recording data to provide justifiable inferences that can be explained with multiple data sources. Future studies may compare against DataSpread, once it is more feature complete.

6 STUDY 1: RESULTS AND TAKEAWAYS

In this section, we analyze the quantitative and qualitative data collected during the quiz and interview phases to address our research questions. Additional details and statistical significance test results can be found in our extended technical report [66].

6.1 Task Performance

In Figure 5a and 5b, we show the distribution of submission times of participants for the five task categories, for birdstrikes and airbnb respectively. For all task categories, *participants' submission times using NOAH were lower than Excel on average.* Moreover, 19 out of 20 participants completed at least four tasks in less time using NOAH. In Figure 6a and 6b, we show the percentage of correct submissions per task category, for birdstrikes and airbnb, respectively. For all tasks except for the fourth, compare (2), for which the accuracy was the same for both tools, participants attained higher accuracy with NOAH compared to Excel. We further evaluated the statistical significance of the completion time and accuracy. For all tasks except customize, the submission time gains with NOAH were statistically significant. However, the improvement in accuracy was significant for the steer task only.

Both results suggest that the capabilities offered by NOAH made spreadsheet exploration faster while also improving task accuracy. However, participants' task submissions with NOAH were not completely error free (Figure 6). Moreover, the submission times for compare (N) and customize tasks with NOAH were comparatively higher than other task categories. Next, we explore the pros and cons of NOAH's features based on participant feedback and our analysis of the video recordings of the study sessions.

Participant preferences. Our analysis of the survey results revealed that participants preferred NOAH to Excel. On a scale of 1 to 7 (7 highest), participants found NOAH to be easier to use (5.88), a faster means for navigation (6.03), and easier to comprehend (5.60) than Excel (rated 4.33, 4.22, and 4.48, respectively). We discuss the results in detail in the technical report [66].

6.2 Qualitative Feedback

We now assess various components of NOAH impacted participants' navigation performance. We also discuss the benefits (indicated with 👍) and limitations (indicated with 👎).

6.2.1 Dynamic Hierarchical Overview. 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, some of the newer interactions had a steeper learning curve.

👍 **Exploring large datasets.** The hierarchical overview aided exploration of large spreadsheet datasets. Participants found it difficult to perform various navigation tasks in Excel, especially on operations that required perusing data across multiple screens. With NOAH, participants avoided endless scrolling via clicking and semantically zooming on the hierarchical 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 to explore large datasets with NOAH.

👍 **Binning and customization as filtering alternatives.** 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 ($N = 14$) preferred the feature to Excel's filtering feature when working with numeric data. 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.

👎 **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*" ($P11$).

👎 **Exploring categorical data.** 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).*"

6.2.2 Aggregate Column. While the aggregate column feature enabled in-situ steering free computation and eliminated errors for steer tasks, comparing the results of bins across multiple granularities introduced visual discontinuity.

👍 **Easier and more accurate formula computation.** 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*" ($P4$). Multiple participants ($N = 13$) found it easier to issue formulae with the aggregate column feature in NOAH. However, the accuracies and submission times for the steer tasks in Excel varied significantly across datasets (see Figure 5 and 6). All of the 14 inaccurate submissions with Excel involved steering an incorrect spreadsheet region; 11 of the inaccurate submissions were with the airbnb dataset. $P12$ commented: "*With NOAH, you don't have to highlight every number versus Excel where you actually have to select everything.*" For the steer task with the birdstrikes (airbnb) dataset, participants were asked to count the number of cells with value 1 (> 60, respectively) in a given column. Analysis of screen recordings revealed that, for Excel, with birdstrikes, several participants used the *autosum* feature instead of the `COUNTIF` formula to quickly count the 1's in the binary-valued column. Participants avoiding data steering resulted in fewer errors. However, for the airbnb dataset, participants could not use a similar shortcut and had to manually steer the data.

👎 **Visual discontinuity during comparisons.** For the task compare (N), participants had to perform N comparisons in the aggregate column. 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 5). In addition, the accuracies ($N = 13$ accurate) of the compare (N) task in Excel were lower.

6.2.3 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 task compare (2), all of the participants used the context bar in NOAH 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 aggregates 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.

👍 **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 conditional formatting 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 via automatically colored cells, helping them relate the aggregate column with the raw data—"*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*" ($P5$).

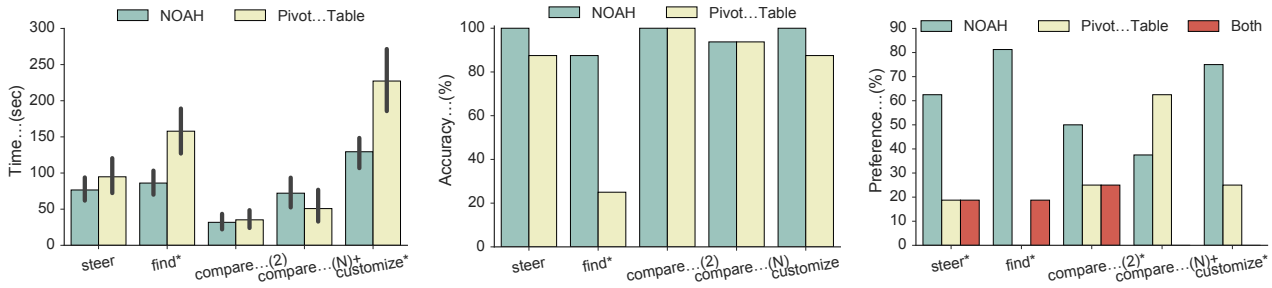
7 STUDY 2: NOAH VS. PIVOT TABLE

The previous study investigates the impact of integrating an overview with spreadsheet systems. We now present another study aimed at characterizing the benefits and limitations of a dynamic hierarchical overview like NOAH compared to Pivot Table, an OLAP-style overview feature available in popular spreadsheet systems. Similar to NOAH, Pivot Table complements spreadsheets, providing a non-hierarchical (flat) overview of the data in a separate area of the spreadsheet while enabling aggregation operations as an alternative to issuing formulae. We opted against comparing NOAH with other BI tools such as Tableau [73] as such tools are not integrated within existing spreadsheet systems. We further discuss the differences in goals between spreadsheets systems and BI tools in Section 8.

7.1 Study Design and Participants

The study was designed to answer the following question: *How does the integration of an interactive hierarchical overview like NOAH impact the usability and efficacy of spreadsheet data exploration compared to non-interactive flat overviews like Pivot Table?*

We conducted the study remotely via the Zoom video communication tool with 16 participants (3 female, 13 male). The two systems used in the study were: the Pivot Table feature within Google Sheets, and NOAH integrated within DataSpread. As we conducted the study remotely, using Google Sheets ensured the consistency of spreadsheet systems across participants with various operating systems. Due to the parity between and similar findings from the two datasets used in the prior study, we selected only one dataset, airbnb, for this study while creating two task sets A and B. Set A tasks were the same as the previous study. Set B tasks were similar to the set A tasks, but the names of the cities were different. We alternated the order of the tools between consecutive participants while keeping the order of the question sets fixed: tasks in set A were assigned to the first tool; tasks in set B were assigned to the second tool. This setting enabled us to obtain more data points (eight) for each set ($= \{A, B\}$) and tool ($= \{$NOAH, Pivot Table$\}$) combination. Before performing the tasks of each set, participants first watched a video tutorial of the tool and then used the corresponding tool on a separate dataset [59] to familiarize themselves with its features. For NOAH we reused the video tutorial from the prior study. For Pivot Table, we provided a video tutorial that covered



(a) Submission Time **(b) Accuracy** **(b) Participant Preference**

Figure 7: Study results (∗ denotes statistically significant improvements for NOAH (α = 5%) relative to Pivot Table and + denotes the opposite).

all of its crucial features like adding rows, columns, aggregation functions, and filters. We then conducted an interview and survey to collect participant's qualitative feedback regarding the tools. We evaluated the accuracy and completion time for the tasks.

7.2 Key Findings

We now analyze the quantitative and qualitative data collected during the quiz, interview, and survey phases.

Overall Task Performance. As depicted in Figure 7a and 7b, *NOAH helped participants achieve higher accuracy and less time overall than Pivot Table, and, in particular, for three out of the five tasks.* Participants' performance for the compare (2) task was similar for both tools. To complete the compare (N) task that required comparing summaries of multiple bins (cities), participants took more time with NOAH than Pivot Table. We explain the reason behind the higher latency with NOAH later. On average, with NOAH participants achieved an accuracy of 96.88%—20.78% more than Pivot Table—in about 28.47% less time. While it is easier to explore a Pivot Table for low cardinality attributes, as the cardinality increases, it becomes difficult to mentally assimilate the overview. The hierarchical overview NOAH makes the information more perceptually scalable. We discuss the benefits and limitations of the tools next.

Benefits of NOAH Over Pivot Table. Participants provided more accurate responses in less time for NOAH were steer, find, and customize tasks. Figure 7c depicts the % of participants that preferred NOAH, Pivot Table, or both for each task. As depicted in the figure, participants' preferences are correlated with their accuracy and completion times: *most participants preferred the (a) ease of issuing formulae and viewing results for the steer tasks, (b) coordinated interactions when looking up related spreadsheet data for the find task, and (c) perceptually scalable and customizable overview when exploring high cardinality values for the customize task.*

👍 **Ease of issuing formula and viewing summaries.** For steer tasks, participants found it easier to issue formulae using aggregate columns in NOAH and then locate the desired results by exploring the overview bins. NOAH prompts the user to provide the necessary parameters based on the formula chosen in the aggregate column menu. Pivot Table required a user to determine and interpret various parameters such as rows, columns, values, aggregate functions, and filters. Some of the parameters like columns were often not required for the task at hand making it cumbersome.

👍 **Easier navigation with coupled interactions.** For the find task, in particular, participants preferred the coordination between the overview and spreadsheet in NOAH—participants clicked on the corresponding bin, *e.g.*, Chicago, to jump to the relevant data in the spreadsheet. Moreover, the auto-highlighting feature in NOAH enabled participants to quickly identify rows that satisfied the condition in the formula. Unlike NOAH, Pivot Table lacks coordination with the spreadsheet. Therefore, participants had to tediously scroll through the spreadsheet to find the desired data and then manually verify which rows satisfied the filtering condition.

👍 **Perceptual scalability with large data.** For the customize task, using NOAH, participants were able to visually inspect the bins and identify which bins required customization, to obtain the desired range. The corresponding Pivot Table spanned multiple pages due to 365 unique values in the availability column—to get an overview of the summary, participants had to tediously scroll through the Pivot Table resulting in visual discontinuity similar to vanilla spreadsheets and contributing to higher delay.

Limitations of NOAH Compared to Pivot Tables. The limitations of NOAH were related to comparing multiple bins and the unfamiliarity with new interactions like bin customization.

👎 **Cumbersome Exploration with Categorical Data.** When working with categorical data, the hierarchical overview of NOAH contributed to higher latency and lower accuracy for the compare (N) task. The task performance was slightly better with NOAH when comparing only two categories for the compare (2) task. The hierarchy introduced visual discontinuity—as all the cities were not visible in a single screen, participants found it difficult to compare the summaries. For the compare (N) task, participants had to first transform the hierarchical overview into a flat overview using a bin customization option called "split all" and then manually compare all the cities. This added step, along with the manual comparison, contributed to the higher latency. Pivot Table's flat overview presentation, on the other hand, eliminated the need for any additional customization. Participants (N = 10) mentioned that such a flat overview was desirable for exploring categorical data—"*With Pivot Table it was easier to view all the results in the same table. [With] NOAH, I had to split all the bins [cities] and then compare. I would have preferred everything [all the cities] automatically split.*" ($P4$).

👎 **Unfamiliarity of Operations.** Some participants (N = 5) preferred Pivot Table due its simplicity in result presentation for categorical data. Participants (N = 4) also mentioned that Pivot Table operations were more relatable to spreadsheets and that the learning curve in NOAH was slightly steeper. For example, Pivot Table can be sorted by the summaries, similar to how a spreadsheet is sorted. However, NOAH provides bin customization as the only overview organization operation and interactions like merging and splitting bins were different from spreadsheet semantics.

Settings Where Each Tool is Preferable. When asked in which scenarios participants might prefer one tool over another, multiple participants ($N = 7$) mentioned that NOAH is preferable when the dataset is large or unknown. Similarly, multiple participants ($N = 9$) suggested Pivot Table is more suitable for working with known datasets with predefined tasks. Participants ($N = 6$) also mentioned that NOAH can be useful for exploratory analysis of spreadsheet data—"*For a dataset [that] I haven not looked at before, I would prefer NOAH to get an intuitive sense of data*" ($P6$). Overall, participants ($N = 13$) preferred the visual representation of the NOAH overview over Pivot Table as it captured the overall data distribution via histograms—"*[I] like the histogram that NOAH generates. The visual cues were helpful in exploring and comparing the results*" ($P16$).

Takeaways. The findings above indicate that NOAH presently works best for open-ended tasks or initial exploration, while Pivot Table works best for settings that are more constrained, *e.g.*, when comparing a small number of categorical values directly. Future versions of NOAH can provide the "best of both worlds" by allowing a single button to toggle between the flat (as in Pivot Table) and hierarchical views, especially for categorical data. 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 hierarchial overview. Similar to Pivot Table, we can provide support for additional operations like sort, filter, and find, on the overview. The bin customization operations, *i.e.*, bin splitting and merging, can be made more similar to how spreadsheet cells are split or merged—in Excel, these operations are direct and only require a single click.

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. Excel users can manually create references to a spreadsheet region using *named ranges* [70] and later, click on a named range to navigate to the referred region. However, the onus is on the user to create these ranges. The Pivot Table [77] and `SUBTOTAL` [52] features allow users to view summary results. We have already compared NOAH with Pivot Table in Section 7. The `SUBTOTAL` function, embeds summaries with the raw data. For datasets with many unique values, this summary view introduces visual discontinuity during exploration, similar to a Pivot Table. Google Sheets Explore [25] summarizes spreadsheet data with auto-generated charts, but doesn't address the challenges related to scrolling and steering.

Enhancing Spreadsheet Scalability. Recent work has shown that users experience spreadsheets crashing and freezing even on millions of rows [43]. Some systems [5, 11, 68], support large spreadsheets beyond main-memory limitations, by using scalable persistent or cloud storage, or approximation. Other work has targeted making present-day spreadsheets more interactive [6, 67]. Our work is complementary—it instead targets users' perceptual challenges in exploring present-day spreadsheet that support fewer than a million rows. That said, NOAH can potentially be extended to support overviews on datasets that go beyond current spreadsheet limits using standard AQP approaches. Other work has targeted trying to make spreadsheets more expressive [2, 3, 7], more robust [4, 29], and more interactive feature-rich [18, 35, 41, 58, 63].

Tabular Data Summarization. Systems like Oracle Service Cloud Analytics [60] and SSRS [51] can *group* tabular data into high-level summaries, while supporting *drill down* capabilities. However, users are required to import the data to a database to use these features. Unlike NOAH, these summaries lack embedded visual cues (*e.g.*, histograms, cell auto-highlights), a feature which many study participants identified as a positive. BI tools such as Tableau [73], Power BI [50], Metabase [46], and Keshif [79] all provide visual summaries of tabular data. Both tabular [51, 60] and visual [46, 50, 73, 79] summaries require users to import the data to a different system. Therefore, these overviews lack coordination with the actual spreadsheet—much like Pivot Tables, these summaries are not dynamically linked to nor are co-located with the spreadsheet. As a result, users lose the ability to directly manipulate raw data, derive new data, and issue formulae for free-form analysis. Therefore, the goals of spreadsheets differ from the tabular data summarization tools in two ways: a) facilitating direct manipulation of raw data in-situ and b) enabling arbitrary derivation of new data and summaries using various operations involving navigation, *e.g.*, issuing formulae. NOAH, being a plug-in to spreadsheets, provides a unified interface that upholds both these goals while enhancing navigational capabilities for spreadsheet users.

Database Usability. There has been a lot of recent interest over the past decade in making databases and data exploration systems more usable [33]. A number of papers have targeted more intuitive and interactive specification interfaces for databases [1, 31, 34, 40, 56]. Other work has targeted query suggestion, wherein queries used by others are used for recommendations [12, 23, 37, 38, 53].

9 CONCLUSION

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 design in NOAH, a spreadsheet navigation plug-in. NOAH employs an approximately equi-depth histogram construction strategy to compute an overview on demand. Using NOAH, 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. Quantitatively, we found that NOAH sped up spreadsheet navigation without compromising accuracy. Qualitatively, study participants identified NOAH as positively impacting their experience while overviewing, navigating, and performing computation on large datasets. Moreover, participants found NOAH to be more suitable than Pivot Tables for exploring large and unknown datasets with many unique values. Finally, we identified several enhancements to NOAH while discussing how the proposed overview plug-in can be integrated into existing spreadsheet tools.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. We also thank Nathan Zhang for help in conducting the second user study and subsequent data collection. We acknowledge support from grants IIS-1940759 and IIS-1940757 awarded by the National Science Foundation, and funds from the Alfred P. Sloan Foundation, Facebook, Adobe, Toyota Research Institute, Google, and the Siebel Energy Institute. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies and organizations.

REFERENCES

[1] Azza Abouzied, Joseph Hellerstein, and Avi Silberschatz. 2012. DataPlay: interactive tweaking and example-driven correction of graphical database queries. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 207–218.

[2] 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.

[3] 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.

[4] 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.

[5] Mangesh Bendre, Vipul Venkataraman, Xinyan Zhou, Kevin Chang, and Aditya Parameswaran. 2018. Towards a holistic integration of spreadsheets with databases: A scalable storage engine for presentational data management. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 113–124.

[6] 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.

[7] 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.

[8] 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.

[9] Matthew Brehmer, Stephen Ingram, Jonathan Stray, and Tamara Munzner. 2014. Overview: The design, adoption, and analysis of a visual document mining tool for investigative journalists. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2271–2280.

[10] 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.

[11] 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.

[12] Ugur Cetintemel, Mitch Cherniack, Justin DeBrabant, Yanlei Diao, Kyriaki Dimitriadou, Alexander Kalinin, Olga Papaemmanouil, and Stanley B Zdonik. 2013. Query Steering for Interactive Data Exploration. In *CIDR*.

[13] Yolande E Chan and Veda C Storey. 1996. The use of spreadsheets in organizations: Determinants and consequences. *Information & Management* 31, 3 (1996), 119–134.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Murray Cox. 2020. The inside airbnb dataset. Retrieved December 15, 2020 from http://insideairbnb.com/get-the-data.html

[21] 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*.

[22] 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.

[23] 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.

[24] Johanna Fulda, Matthew Brehmel, and Tamara Munzner. 2015. TimeLineCurator: Interactive authoring of visual timelines from unstructured text. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 300–309.

[25] Google. 2020. Google Explore. Retrieved December 15, 2020 from https://support.google.com/docs/answer/6280499

[26] Google. 2020. Google Sheets scale. Retrieved December 15, 2020 from https://developers.google.com/drive/answer/37603

[27] 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.

[28] 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.

[29] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* 46, 1 (2011), 317–330.

[30] Thomas Höllt, Nicola Pezzotti, Vincent van Unen, Frits Koning, Elmar Eisemann, Boudewijn Lelieveldt, and Anna Vilanova. 2016. Cytosplore: Interactive Immune Cell Phenotyping for Large Single-Cell Datasets. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 171–180.

[31] Stratos Idreos and Erietta Liarou. 2013. dbTouch: Analytics at your Fingertips.. In *CIDR*.

[32] Yannis Ioannidis. 2003. The history of histograms (abridged). In *Proceedings 2003 VLDB Conference*. Elsevier, 19–30.

[33] HV Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. 2007. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 13–24.

[34] 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).

[35] 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.

[36] Miranda Katz. 2017. A Lone Data Whiz Is Fighting Airbnb—and Winning. Retrieved December 15, 2020 from https://www.wired.com/2017/02/a-lone-data-whiz-is-fighting-airbnb-and-winning/

[37] Nodira Khoussainova, Magda Balazinska, Wolfgang Gatterbauer, YongChul Kwon, and Dan Suciu. 2009. A Case for A Collaborative Query Management System.. In *CIDR*.

[38] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. 2010. SnipSuggest: Context-aware autocompletion for SQL. *VLDB Endowment* 4, 1 (2010), 22–33.

[39] 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.

[40] 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.

[41] Marc Levoy. 1994. Spreadsheets for images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 139–146.

[42] 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.

[43] 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*.

[44] Jock D. Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1137–1144. https://doi.org/10.1109/TVCG.2007.70594

[45] Liam McNabb and Robert S Laramee. 2017. Survey of Surveys (SoS)-Mapping The Landscape of Survey Papers in Information Visualization. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 589–617.

[46] Metabase. 2020. Metabase Inc. Retrieved December 15, 2020 from https://www.metabase.com/

[47] Miriah Meyer, Michael Sedlmair, P Samuel Quinan, and Tamara Munzner. 2015. The nested blocks and guidelines model. *Information Visualization* 14, 3 (2015), 234–249.

[48] Microsoft. 2020. Excel Specifications and Limits. Retrieved December 15, 2020 from https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3

[49] Microsoft. 2020. How finance leaders can drive performance. Retrieved December 15, 2020 from https://enterprise.microsoft.com/en-gb/articles/roles/finance-leader/how-finance-leaders-can-drive-performance/

[50] Microsoft. 2020. Power BI. Retrieved December 15, 2020 from https://powerbi.microsoft.com/en-us/

[51] Microsoft. 2020. SQL Server Reporting Services. Retrieved December 15, 2020 from https://docs.microsoft.com/en-us/sql/reporting-services

[52] Microsoft. 2020. SUBTOTAL. Retrieved December 15, 2020 from https://support.office.com/en-us/article/subtotal-function-7b027003-f060-4ade-9040-e478765b9939

[53] 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.

[54] 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.

[55] 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.

[56] 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

[57] Bonnie A Nardi and James R Miller. 1990. *The spreadsheet interface: A basis for end user programming*. Hewlett-Packard Laboratories.

[58] Fabian Nunez and Edwin Blake. 2000. Vissh: A data visualisation spreadsheet. In *Data Visualization 2000*. Springer, 209–218.

[59] Bureau of Transportation Statistics. 2020. The US flight dataset. Retrieved December 15, 2020 from https://www.transtats.bts.gov/

[60] Oracle. 2020. Oracle Service Cloud Analytics. Retrieved December 15, 2020 from https://docs.oracle.com/cloud/may2015/servicecs_gs/servicecs_report.htm

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

[62] 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.

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

[64] Johannes Pretorius, Helen C Purchase, and John T Stasko. 2014. Tasks for multivariate network analysis. In *Multivariate Network Visualization*. Springer, 77–95.

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

[66] Sajjadur Rahman et al. 2020. NOAH: Interactive Spreadsheet Exploration with Dynamic Hierarchical Overviews. *http://tiny.cc/noahTR* (2020).

[67] 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.

[68] Vijayshankar Raman, Andy Chou, and Joseph M Hellerstein. 1999. Scalable Spreadsheets for Interactive Data Analysis. In *ACM SIGMOD Workshop on DMKD*.

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

[70] Named ranges. 2020. Define and use names in formulas. Retrieved December 15, 2020 from https://www.excel-easy.com/examples/names-in-formulas.html

[71] 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.

[72] 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.

[73] Tableau. 2020. Tableau Software. Retrieved December 15, 2020 from https://www.tableau.com/

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

[75] 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.

[76] 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.

[77] Wikipedia. 2020. Pivot Table. Retrieved December 15, 2020 from https://en.wikipedia.org/wiki/Pivot_table

[78] 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.

[79] 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.

[80] 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)*.