# Towards Interactive, Navigable, and Expressive Spreadsheets: DATASPREAD to the Rescue!

Mangesh Bendre, Kelly Mack, Sajjadur Rahman, Tana Wattanawaroon,
Yan Liu, Yu Lu, Ping-Jing Yang, Stanford Zhou, Xinyan Zhou
Kevin Chen-Chuan Chang, Karrie Karahalios, Aditya Parameswaran
University of Illinois at Urbana-Champaign (UIUC)
{bendre1 | knmack2 | srahman7 | wattana2 | yanliu8 | yulu3 | py2 | szhou3 | xzhou14 |
kcchang | kkarahal | adityagp}@illinois.edu

## ABSTRACT

Spreadsheet tools are ubiquitous for interactive ad-hoc tabular data management, especially for small-to-medium datasets. With increasing dataset sizes, spreadsheet tools fall short—they crash or hang due to heavy computation embedded within the sheet (interactivity); they don't support effortless navigation to desired areas when spreadsheets go beyond a certain size, and scrolling becomes ineffective (navigability); they only support simplistic cell-at-a-time computation, severely limiting the power of computations over collections of data (expressivity). On the other hand, relational database systems offer substantial benefits in terms of scalability and expressivity over spreadsheet tools, but lack the ease-of-use and flexible analysis capabilities. Over the past four years, we have been developing DATASPREAD: a tool that holistically unifies databases and spreadsheets to leverage the benefits of both, with a spreadsheet-like front-end and a database-like backend. In this paper, we demonstrate the current state of DATASPREAD, along with three key features that we have developed to address the aforementioned spreadsheet scalability challenges in interactivity, navigability, and expressivity. Our demonstration will let attendees perform typical data analysis tasks on Microsoft Excel and DATASPREAD side-by-side, providing a clear understanding of the improvements offered by DATASPREAD for large datasets over spreadsheet tools.

## 1. INTRODUCTION

Spreadsheet tools are commonly used for interactive analysis and management of tabular data by novice and expert users alike. From personal bookkeeping to complex financial reports, scientific data analysis to scheduling, the ubiquity of spreadsheets as a data management tool is unparalleled [1]. The main advantages offered by spreadsheet tools, *e.g.*, Excel, Google Sheets, and LibreOffice, include the ability to directly manipulate data, an intuitive user interface, and a flexible data model with the ability to add new rows, columns, and tuples seamlessly. However, the sheer scale of data available in many domains has started to expose several limitations of traditional spreadsheet tools, making them ineffective at manipulating and interacting with large spreadsheets. Our recent study of scalability issues in spreadsheet tools [2] via analysis of Reddit posts [3] revealed 83 separate accounts of issues with Excel that were, at least in part, due to the scale of data or operations on the spreadsheet. Anecdotal evidence also indicates that users struggle with a variety of scalability problems on large and/or complex spreadsheets. To counter this, spreadsheet tools curb the size of data being analyzed, *e.g.*, Excel (Google Sheets) imposes a 1M (2M) limit on the number of rows (cells). Even so, many of the Reddit posts indicate scalability issues arising with tens of thousands of rows, well below the million row mark [2].

**Limitations of current spreadsheets.** We identify three key shortcomings of traditional spreadsheet tools related to the scale of the data, using Microsoft Excel as a concrete example. *(a) Interactivity.* Excel ceases to be interactive when dealing with computationally heavy spreadsheets. One user posted on Reddit that complex calculations on Excel can take as long as four hours to finish during which the user interface is unresponsive: *"approximately 90% of the time I spend with the spreadsheet is waiting for it to recalculate"*[1]. *(b) Navigability.* While Excel supports basic browsing, its tabular representation of data does not support quick navigation to desired areas of the spreadsheet, or provide a high-level understanding of the data distribution within a spreadsheet. One Reddit user commented: *"Inexperienced Excel users are unable to navigate data efficiently"*[1]. *(c) Expressivity.* Excel uses a cell-at-a-time-based formula query model—this makes it cumbersome to express the much more convenient and powerful table-oriented (or relational) operations when working with tabular data. For example, to filter records that occur between two dates, one of the Reddit users suggested the following Excel formula: IFERROR(INDEX(A:A, SMALL(IF(($A$20:$A$1000 >=$E$40) * ($A$20:$A$1000<=$F$40),ROWS ($A$20:$A$1000)), ROWS ($C$5:$C5))),"")[1], which is a cumbersome way to express the simple relational operation.

**Contributions.** Our tool, DATASPREAD, is designed to be a *one-stop tool that unifies the capabilities of spreadsheets and databases to provide an intuitive direct-manipulation* [4] *interface for managing big data*. DATASPREAD addresses the above limitations, making it interactive, navigable, and expressive while working with large spreadsheets, with the following features: *(a)* an *asynchronous and lazy computational model* to address the issue of poor interactivity; *(b)* a *navigation interface* to enable users to drill-down to desired areas while examining a summarized view of the data to improve navigability; *(c) support for table-oriented formulae*, a simple but effective means to express relational operations on tabular regions to improve expressivity. For example, the complex formula described earlier can be expressed using the following table-oriented formula: SELECT(A20:A1000, ATTR_DATE $\leq$ F40 && ATTR_DATE $\geq$ E40), which is cleaner and easier to understand than the INDEX function.

**Challenges.** Developing the aforementioned features presents a host of engineering and research challenges, ranging from storage and indexing to interface usability. First, maintaining a balance between interactivity and consistency of the asynchronous, lazy computation model requires us to compactly encode the dependencies

[1] All Reddit quotes are paraphrased to preserve anonymity.

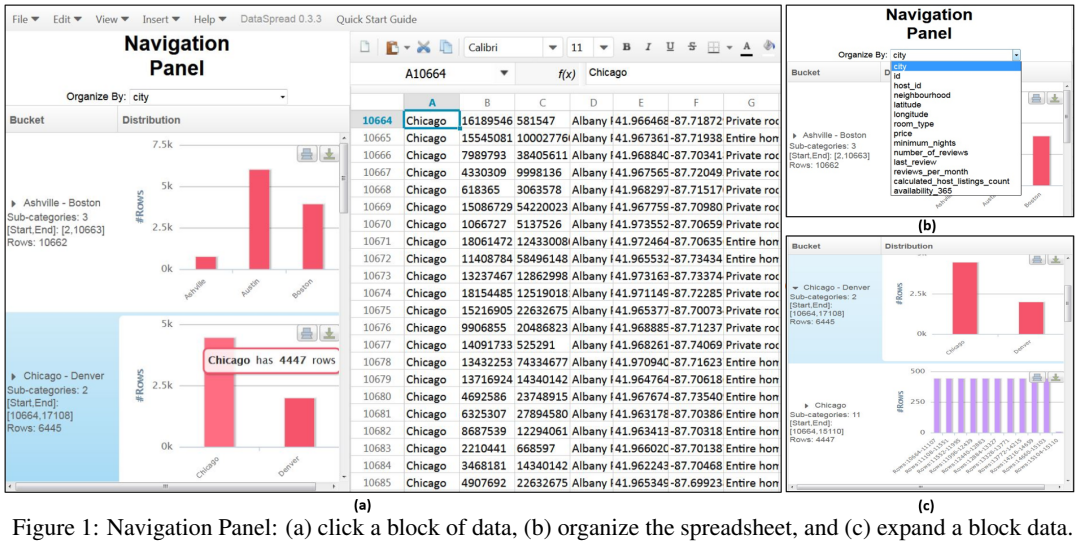


Figure 1: Navigation Panel: (a) click a block of data, (b) organize the spreadsheet, and (c) expand a block data.

across formulae to "hide" cells that need recomputation, as well as schedule computation in a way that takes advantage of shared context and locality. Second, seamlessly integrating the navigation interface within the spreadsheet ecosystem introduces design challenges in both the data-structure that can capture changing orders as a first-class citizen, and simultaneously provide summarized representations of the data. Third, introducing table-oriented formulae akin to relational algebra requires careful design to ensure consistency with the cell-at-a-time model of Excel, and the fact that the multi-cell results of table-oriented formulae, when left unchecked, can "overwrite" other data in the spreadsheet.

Beyond these challenges for the three features, we have also encountered a number of other challenges in developing DATASPREAD to holistically integrate spreadsheet and database capabilities. To go beyond main-memory limitations (*e.g.*, 1M rows), DATASPREAD fetches data *on-demand* from the underlying database when triggered by a user action (like scrolling) or from a system action (like calculating a formula), but identifying ideal representation schemes for spreadsheet data, encompassing the variety of structures typically found in spreadsheets, is not straightforward (and is in fact intractable) [5]. Moreover, we require "positional" data structures that can retrieve data required for specific computations, while not being sensitive to operations that change the position of most of the data items on the spreadsheet (*e.g.*, inserting a row at the start).

**Status.** DATASPREAD is a scalable web application with PostgreSQL as the relational database back-end, and a ZK (open-source) spreadsheet front-end [6]. All the screenshots that we depict in this paper are from our current prototype. The prototype along with its source code, documentation, and user guide can be found at http://dataspread.github.io.

DATASPREAD, is a multi-year project currently in its fourth year of development. Our demo papers serve as milestones for the project. In our earlier demo [7], we described our first attempt at integrating the traditional spreadsheet interface with a relational data store to persist data, demonstrating a proof-of-concept implementation. In this demo, we demonstrate a scalable web-based prototype that introduces features that enhance the user experience beyond what traditional spreadsheets provide with a goal to enable the users to work with large spreadsheets efficiently.

**Related Work.** A number of recent papers have attempted to enrich spreadsheets and relational databases with features from one another via three orthogonal directions: *(a)* Use of spreadsheets to mimic database functionalities [8]. While this approach achieves higher expressivity (via SQL), it is unable to leverage the scalability of databases. *(b)* Use of databases to mimic spreadsheet functionalities [9, 10]. While this approach provides scalability via databases, it is does not support flexible tabular data management as in spreadsheets. *(c)* Use of spreadsheets for querying data [11] and reorganizing data [12]. The former provides an intuitive interface to query data, but loses the expressivity of SQL as well as ad-hoc data management capabilities. The latter proposes novel interface interactions on nested data that enable users to perform grouping, sorting and join operations. These techniques only apply to small scale hierarchical data and cannot be generalized.

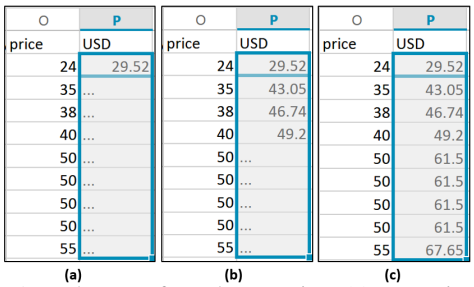


Figure 2: Asynchronous formula execution: (a) user writes formula and make copies by dragging the autofill handle, (b) display partial results, and (c) formula execution completed.

## 2. DATASPREAD OVERVIEW

We now discuss in-depth the new features we developed to address the shortcomings of traditional spreadsheet tools with respect to interactivity, navigability, and expressivity and the challenges we solved along the way.

### 2.1 Asynchronous Computational Model

In Excel, each change the user makes (*e.g.*, changing values or formulae), triggers a sequence of recomputation of formulae, which may take minutes to complete, depending on the size of the data. Excel only returns control to the user when the computation is complete, adopting a *synchronous computation model*—here, the user is kept waiting until the control returns, disrupting interactivity. DATASPREAD adopts an *asynchronous computation model* instead, returning control back to the user immediately, while masking the "dirty" cells (*i.e.*, those whose values have not been computed yet) using ellipsis ("...") on the interface (see Figure 2), and computing

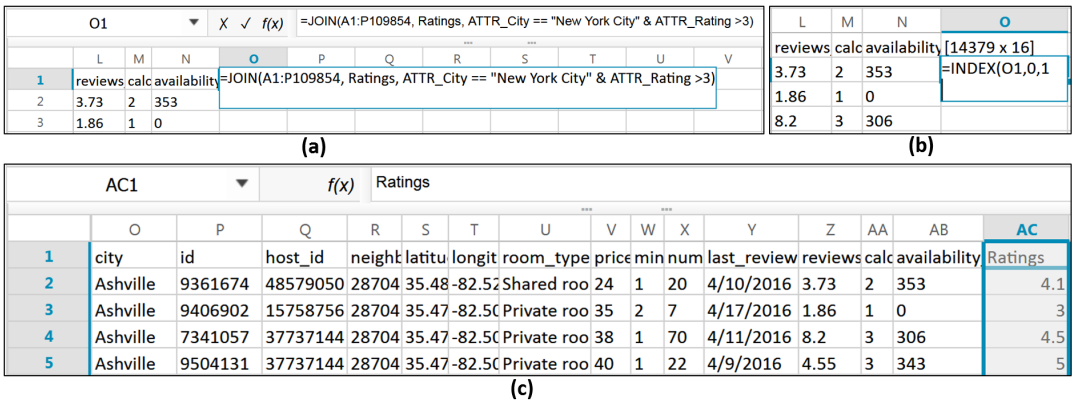


Figure 3: JOIN operation in DATASPREAD: (a) user writes the JOIN formula, (b) user writes the INDEX formula to retrieve JOIN results, and (c) results displayed.

them lazily in the background, exploiting shared computation, and prioritizing for what is seen over what is not seen. Thus, this model ensures interactivity by bounding the time for which the system remains unresponsive after an update.

Instead of masking the "dirty" cells, a simpler approach would be to display the current (stale) values of these cells, and not mark them inaccessible with a ". . ." on the interface. However, this approach can confuse the user by showing them inconsistent data, hence one of our goals is to ensure consistency of data shown to the user at all times. Thus, we adopt the approach of using ". . ." to indicate cells whose values have not been computed yet.

**Challenges.** The primary challenge for enabling asynchronous computation is to maintain consistency and interactivity at the same time. This requires addressing two problems both of which are NP-HARD: identifying the impacted cells after an update on a spreadsheet in a bounded period of time, and determining how to compute the impacted cells. To quickly return control of the spreadsheet back to the user, we also require indexing mechanisms that can support positional (*i.e.*, row and column number based) access, which is required by the formulae embedded within spreadsheets.

**Insight.** Spreadsheet formulae introduce dependencies between different cells on a spreadsheet, which DATASPREAD captures via a *dependency graph*. The dependency graph can tolerate false positives, *i.e.*, identifying a region as being impacted by an update, even when it is actually not, and can be compressed lossily; false negatives are not permitted, as they violate consistency. We develop a greedy compression algorithm to tackle the NP-HARD problem of dependency graph compression to minimize false positives. Using this lossily compressed graph, DATASPREAD can identify the impacted cells in a bounded time, ensuring interactivity. Scheduling the computation of the impacted cells is also an NP-HARD problem. We implement a variant of the weighted shortest job first problem [13] to compute the dirty cells efficiently in a cache-friendly manner and prioritize visible cells, thereby minimizing the time that users see dirty cells. These execution algorithms are aided by novel data models that minimize the amount of data accessed, as well as positional indexing mechanisms that allow rapid access of data by position, as described in our recent ICDE paper [5].

## 2.2 Navigation Interface

Say a user wants to access a specific area of interest within a spreadsheet. At present, the user would have to resort to scrolling to skim over the spreadsheet to arrive at the desired area, which can be painful if the spreadsheet is large. Thus, present spreadsheet tools such as Excel lack a navigation interface. In DATASPREAD, we have developed a navigation interface, which presents a hierarchical view of the spreadsheet, thereby providing an interactive and effective alternative to basic spreadsheet operations such as scrolling and filtering. DATASPREAD's navigation interface organizes and summarizes the spreadsheet so that users can skip over irrelevant regions and access the desired area via simple clicks as opposed to scrolling endlessly (see Figure 1a).

**Challenges.** The primary challenge in introducing such an interface alongside the traditional interface is to seamlessly integrate both interfaces: this integration should allow users to effortlessly perform interactions on both interfaces, enabling rapid interactive exploration and drill-down. In addition, we need a data structure that satisfies the requirements of such an interactive navigation interface, *e.g.*, for dynamic reordering and summarization of data. As the scale of the data grows, maintaining the data structure such that the navigation interface remains interactive is an added challenge.

**Insight.** Similar to the idea adapted by online maps, *e.g.*, Google Maps, the hierarchical navigation interface abstracts the tabular data at different levels of granularity, where users can freely move across different granularities. To enable seamless integration of the navigation interface with the spreadsheet data, we leverage our earlier work on hierarchical positional indexes [5] for tabular data on a spreadsheet. Each level in the positional index maps to a corresponding level in the hierarchical navigation interface. At the lowest level of hierarchy, we display the raw spreadsheet data. At each level of the hierarchy, DATASPREAD abstracts the spreadsheet by a group of blocks—the grouping is determined based on the distribution the data. Each block contains aggregated information corresponding to the spreadsheet region it spans. Each block contains the block name, number of rows, range of the rows, and a histogram depicting the distribution of the corresponding data (see Figure 1a). Users can get an overview via the navigation interface (Figure 1a) and organize the data by different attributes (Figure 1b). The blocks on the interface are shortcuts that enable users to quickly jump to areas of interest within the spreadsheet.

## 2.3 Table Oriented Formulae

Present spreadsheet tools such as Excel do not support computation that go beyond the cell-at-a-time metaphor: for example, relational algebra operations such as general joins are not permitted or are at least not straightforward (*e.g.*, VLOOKUPs for key-foreign key joins). In DATASPREAD, we aim to support both SQL [7] as well as general-purpose relational computation via *table-oriented formulae*, supporting operations such as joins, on both tables from the underlying database or spreadsheet regions, *e.g.*, A3:D4, which are treated as tables. Table-oriented formulae retain the semantics of typical formulae on spreadsheets, while also empowering users

to use relational primitives.

**Challenges.** The primary challenge is to ensure that the table-oriented formulae work seamlessly with the cell-at-a-time formula model of Excel in spite of the differences in ideologies and semantics of spreadsheets and databases. Specifically, a table-oriented formula can return multiple records, which makes it incompatible with the standard spreadsheet formula semantics of returning results in a single cell. Returning multiple records is also problematic because unless we are careful, these records can overflow and overwrite other data present on the spreadsheet.

**Insight.** We support table oriented operations via the following spreadsheet functions: UNION, DIFFERENCE, INTERSECTION, CROSS-PRODUCT, JOIN, FILTER, PROJECT, and RENAME. These functions return a single composite table value representing the tabular result of an operation, but this value is not displayed in the cell. We instead show the dimensions of the composite value (similar to matrix dimensions). To retrieve the individual rows and columns within the tabular result encoded as the composite value, we have an INDEX(cell, i, j) function that looks up the (i, j)-th row and column in the composite table value in location cell, and places it in the current location. By forcing users to use INDEX to look up entries in the composite table value, we can avoid the issue of overflow of records, since only the cells that use an INDEX formula will ever refer to data corresponding to the tabular result. Since the input and output of all these functions is a table, the functions can be arbitrarily nested to obtain complex expressions. So for instance, to do a union of three tables, the user would use UNION(table1, UNION(table2,table3)). Alternatively, users can issue SQL queries on DATASPREAD using the SQL(query, [param1], ...) function, where the first parameter is the SQL query, and the further parameters replace ?'s within the query string. The SQL function is directly executed by the underlying database.

# 3. DEMONSTRATION DESCRIPTION

In our demonstration, conference attendees will interact with DATASPREAD on a dataset from Airbnb [14], utilizing the new user interface constructs, and contrast their user experience with Microsoft Excel. The Airbnb dataset contains publicly available information about Airbnb listings (*e.g.*, listing type, location, reviews, price, availability) of different cities across the world and has ≈570k rows and 16 columns. Our demonstration scenario will simulate the experience of a journalist analyzing the rent price distribution of Airbnb listings using spreadsheets. Attendees will examine the following features while exploring the Airbnb data, contrasting Excel and DATASPREAD.

**Feature 1: Asynchronous computation.** Suppose the journalist finds out that the rent price of listings in European cities are in € and she wants the price in $. She creates a new column *USD* in the spreadsheet with appropriate formulae to convert the "rent price" to $. The journalist then wants to update the $ to € conversion rate. This impacts all of the cells in the USD column of the sheet. Attendees will perform this operation on Excel and DATASPREAD and contrast their experience. As soon as the attendees update the cell on Excel, the interface will be unresponsive until the computation is complete. On the other hand, the attendees can perform this update in DATASPREAD interactively; *i.e.*, the control of the sheet is returned back to the user almost immediately. They will notice that some of the impacted cells show ellipses (. . .), meaning they are under computation 2. The ellipses will be computed in the background and eventually, will be replaced with correct values.

**Feature 2: Navigation Interface.** Suppose the journalist now wants to compute the average price of the listings in Chicago. We will ask the attendees to perform similar operations. With Excel, the attendees have to first sort the data by "City" and then filter out the listings belonging to Chicago and finally, compute the average by providing the range of the Chicago listings to the AVERAGE function. Attendees will witness how tedious this approach can be for large spreadsheets as it involves scrolling through thousands of rows or move up and down the scrollbar to find the range of the Chicago listings. DATASPREAD, on the other hand, groups multiple cities based on the alphabetical order to form the highest level of granularity. Figure 1a shows two such groups: Ashville-Boston and Chicago-Denver. The attendees can utilize this navigation interface to quickly jump to the region of interest by clicking a block. For example, in Figure 1a, clicking on the Chicago-Denver block (highlighted in light blue), enables users to quickly navigate to the corresponding region. Users can drill down further by expanding a block (circled in orange in Figure 1c) if required. Using the range information of the region of interest, the attendees can then calculate the aggregate value..

**Feature 3: Table-oriented Formulae.** Assume that our journalist finds another data set that contains average ratings of all the listings in Airbnb. Suppose the journalist wants to see all the listings in NYC that has average rating above some threshold along with their ratings. We will provide a "Ratings" spreadsheet to the attendees and ask them complete the task. The task corresponds to a *natural join*, which is not possible natively in Excel. The closest solution is an *outer join*, which can be done by a combination of VLOOKUP and IF statements which must then be applied to multiple listings by dragging the autofill handle. Using DATASPREAD, the attendees can use the following two formulae to complete the task: JOIN(A1:P142857, Ratings, City=="NYC" & Rating $>$ 3) and INDEX(O1,0,1) (see Figure 3). Note INDEX can be applied to multiple cells by dragging the autofill handle, just like traditional spreadsheet formulae.

In addition to demonstrating the above features, we will describe how DATASPREAD internals help avoid the pitfalls of traditional spreadsheet tools. Overall, the aforementioned demonstration scenarios will convince attendees that our DATASPREAD system offers an interactive, navigable, and expressive spreadsheet software that enables users to effectively work with large spreadsheets.

# 4. REFERENCES


[1] Spreadsheet software: top five on the market. https://www.techradar.com/news/software/business-software/spreadsheet-software-top-five-on-the-market-1257738, 2015.

[2] K. Mack et al. Characterizing scalability issues in spreadsheet software using online forums. In *SIGCHI*, 2018.

[3] Microsoft Excel spreadsheet subreddit. https://www.reddit.com/r/excel/.

[4] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

[5] M. Bendre et al. Towards a holistic integration of spreadsheets with databases: A scalable storage engine for presentational data management. In *ICDE*, April 2018.

[6] ZK Spreadsheet. https://www.zkoss.org/product/zkspreadsheet.

[7] M. Bendre et al. Dataspread: Unifying databases and spreadsheets. volume 8, pages 2000–2003. VLDB Endowment, Aug. 2015.

[8] J. Tyszkiewicz. Spreadsheet as a relational database engine. In *SIGMOD*, pages 195–206. ACM, 2010.

[9] A. Witkowski et al. Spreadsheets in rdbms for olap. In *SIGMOD*, SIGMOD '03, pages 52–63, New York, NY, USA, 2003. ACM.

[10] A. Witkowski et al. Query by excel. In *VLDB*, pages 1204–1215. VLDB Endowment, 2005.

[11] B. Liu and H. Jagadish. A spreadsheet algebra for a direct data manipulation query interface. In *ICDE*, pages 417–428. IEEE, 2009.

[12] K. S.-P. Chang and B. A. Myers. Using and exploring hierarchical data in spreadsheets. In *SIGCHI*, pages 2497–2507. ACM, 2016.

[13] J. Blazewicz et al. *Scheduling computer and manufacturing processes*. springer science & Business media, 2013.

[14] M. Cox. Inside Airbnb. http://insideairbnb.com/get-the-data.html.