A Tool for Visualizing Patterns of Spreadsheet Function Combinations

Justin A. Middleton
North Carolina State University
Raleigh, North Carolina

Abstract—Spreadsheet environments often come equipped with a plethora of functions to manipulate and calculate data, but it can be difficult to understand how end-users employ these functions in practice. Without this knowledge, both researchers and practitioners lack information about how end users construct sophisticated programs from these basic elements. We developed a tool that visualizes patterns of how functions are combined into formulae within Excel spreadsheets. Using the Enron spreadsheet dataset as an example, this paper shows how the tool can display both common and anomalous formulas and their respective contexts in an actual workbook.

I. INTRODUCTION

Business and research alike owe their debts to spreadsheets, the table-based interface which empowers users to organize and manipulate huge bodies of data [citation for definition]. Their allure is in their versatility: while the novice end user can work without a deep knowledge of programming, the expert can expedite their work with a variety of built-in operations, or functions.[1] As such, it should be of little surprise when Scaffidi and colleagues estimated that by 2012, over 50 million U.S. workers could be using them, including the 25 million who would be writing programs out of the functions included (Scaffidi citations).

Considering this ubiquity, it's crucial to get spreadsheets right. Our failures in their use can be ruinous, as in 2012 when the influential findings on economic growth were reversed by a selection error [Enron paper], among other horror stories.[2]

Fortunately, the vanguards of spreadsheet research have assembled, organized, and released a number of spreadsheet corpora to inform work on how people actually use these tools in different contexts. Collections like EUSES[yada], FUSE[yada], and the Enron corpus[yada] have already enabled fruitful work across the field: [WORK PENDING MORE RESEARCH]. A number of these studies focus on discovering which built-in functions are the most common (enabling something).

A tool, then, which empowers users of many intentions to explore these datasets would be rife with potential. On a simple level, the tool could provide concrete statistics on which functions are used in practice most (or least) often, and what other functions that work in conjunction with them. These numbers, then, could inform future work done on

intelligently recommending functions to users or generating new formulas from these patterns of function use. Spreadsheet APIs, likewise, could be augmented or pruned through the discovery of frequent combinations or anomalous use of functions. Furthermore, if the tool maintains the connection between patterns and the actual, it could serve as a boon to educators as well, guiding lesson plans to the most commonly employed functions and offering a bounty of instructive (and real) examples.

This paper presents such a tool, informed by such varied sources, that visualizes not how people employ individual Excel functions but how they combine to make more sophisticated spreadsheet formulas.

II. RELATED WORK

Spreadsheets, being a core component of business that they are, have prompted the development of many tools to evaluate them and their many qualities. [RESEARCH ON NON-VISUAL ANALYSIS TOOLS]

Furthermore, this tool comes from a line of spreadsheet visualization tools before it, each with a different focus. For example, Breviz, developed by Hermans and colleagues, uses visualization to illustrate workbook dataflow to address the need for a clearly communicable representation of spreadsheets as they move from person to person.

Other studies focus on API and built-in function use outside the domain of spreadsheets. [BUT FIRST I HAVE TO FIND THEM]

III. APPROACH

A. Goals

In visualizing the spreadsheet data, I outlined a few core goals for what the tool should accomplish:

1 **Draw an interactive interface to explore observed function combinations.** The combination space for all Excel functions is massive, let alone the space for observed formulas. Working from data with actual referents in practice, the tool must aid the user in navigating this space.

2 **Emphasize the quantitative patterns in formula construction.** The tool, accommodating datasets of a few spreadsheets to millions, should address the questions of how often the end users employed a certain function and where they used it. In this way, it must show precise

---

[1] https://support.office.com/en-us/article/Excel-functions-alphabetical-b3944572-255d-4efb-bb96-c6d90033e188

[2] http://www.eusprig.org/horror-stories.htm

metrics, such as frequency of use and depth of function nesting, of the dataset it conveys.

3 **Promote a qualitative understanding of the patterns.** Lest these patterns remain abstract, the tool should supplement its observations with concrete instances of relevant formulas from the corpus. Ideally, it should even direct users to the exact cell in the spreadsheets where the formula was used, contextualizing the functions.

Likewise, to bound our scope, we outlined a few goals to specifically avoid accomplishing with this:

!1 **Do not try to directly explain what a function does.** Though the tool tries to foster tries understanding by linking pattern to example, it won't provide a precise description of what a function accomplishes. The tool's user must infer this.

!2 **Do not create new formulas.** This is essentially an exploratory tool, not a generative one. It should not produce any information other than new views of the original data.

!3 **Do not visualize individual formulas.** Though there is room to explore the place of a single function in the group, the tool must design the core visualization around the entire body of data, not the other way around.

B. Description

To show how this all came together and demonstrate the interface, we will use an example of a quick task: discovering what kind of conditions people use in IF statements. Considering only formulas in which the IF function itself is the top-level function (that is, not nested within another function), we will approach a tree that begins like this:
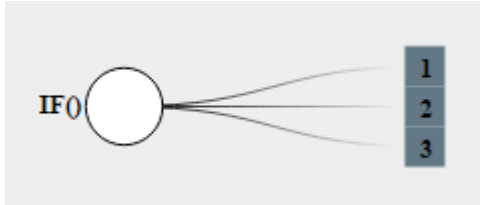

The visualization so far comprises two types of node: the circle, which represents a discrete function in the formula and is size according to its relative frequency (the top-level node, by definition, will be the largest); and the numbered squares, which represent the positions of arguments within its parent function. From this, we can infer that, of the times it was observed, IF can have at most three arguments passed into it, which corresponds with its specifications in the API.[3]

Knowing that it is the first argument which contains the conditionals, we click on the square labeled "1" to explore. To save space, when more than 10 unique arguments have been observed in any position, the tool displays only the first ten, with an option to display the rest. The results are shown below:

[3]https://support.office.com/en-us/article/IF-function-69aed7c9-4e8a-4755-a9bc-aa8bbff73be2
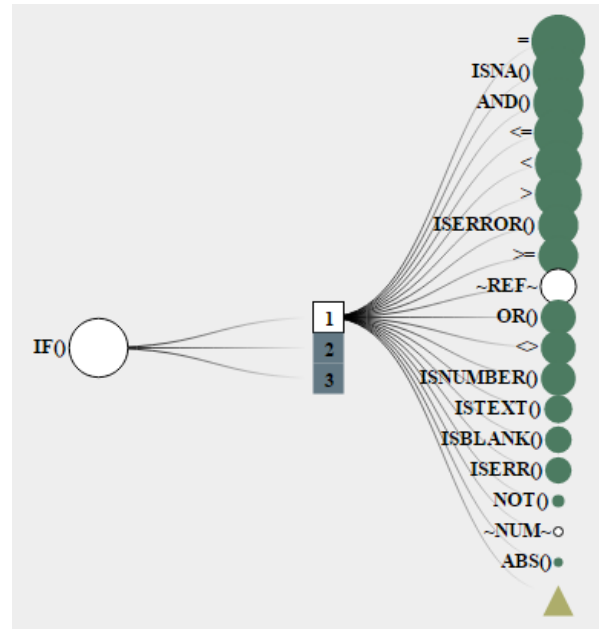
As expected, we see that IF contains as its first argument a number of comparison operators, such as = and ¡=, and boolean-returning functions, like ISNA and AND, with simple equality being the most common and some use as ABS being the least seen of everything actually used. From here, we can further explore the common options among these functions. Clicking on the "=" node will yield two arguments, it being a binary operator, and expanding each of them will peer into the range of common values of equality comparison:
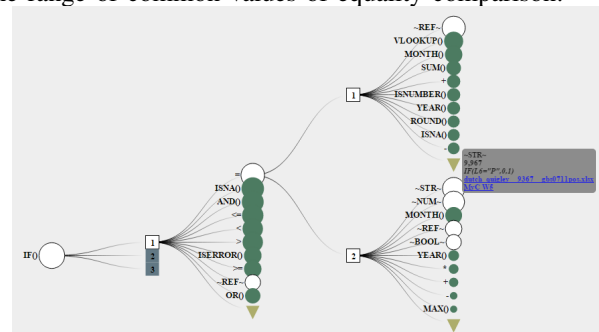
For both sides of the equal sign, the operator has certain types of arguments which predominate over the others: on the left side is most often a reference to another cell; on the right, a string or number literal, which makes sense for the case of confirming a value in another cell before assigning this one. Furthermore, if the concept is difficult to imagine is practice, a tooltip accompanies each function node in the tree, providing a concrete example of a function that uses this structure and where it can be found.

C. Design

Early in the process, we decided that the tree form would be a suitable fit to represent the parent-child or caller-callee relationships inherent in the data, given the composition of formulas as functions and their arguments (which could be yet more functions). By adapting this structure to accommodate a broad range of possibilities for nested functions, the branching

factor depends on both the number of arguments in a function and the number of possible functions observed as an argument in a function.

Guided by the goals in section A, we faced a number of decision points, which we describe below, before we arrived at the design shown above.

1a *Copied formulas*: Excel allows users to spread a formula over an area, repeating the same task in each cell with minor adjustments. Without checking for this, the analysis may not reveal the functions most commonly used together but rather the formulas most often applied to large areas. To combat this, we converted formulas from their native A1 format to the relative R1C1, in which copied formulas should be identical, and reduced the records to unique R1C1 formulas per sheet.

1b *Optional arguments*: How should the tool handle functions which accept a variable number of arguments? SUM, for example, can have anywhere from 1 to 255, and IF can accept either 2 or 3. [4] Without prior evidence, it's possible that spreadsheet programmers use different techniques for different numbers arguments. To account for this, we separate and analyze the different quantities of arguments observed for each function; the tool, however, uses as default all of the options collapsed into a single representation.

2a *Importance of depth*: When a function appears within another, should it be analyzed only as a nested function or would it also be valid to analyze the nested function on its own? If the former, then information about the same function will be scattered across different trees with no way to aggregate them. If the latter, then the tool will analyze some functions multiple times to capture every possible level of nesting. Both approaches have benefits and drawbacks, and so we included both.

2b *Pattern density*: How should the tool quantitatively order its elements: by a function's raw frequency or by the unity of patterns it leads to? For example, if SUM has for its first argument two possibilities, one which itself contains 1000 unique argument possibilities with 1 occurrence each (high frequency, low pattern density) and another seen with 2 argument possibilities of 100 occurrences each (low frequency, high pattern density), which would be more interesting to emphasize? The answer depends on the nuances of the questions, but for simplicity, I've shown the former.

2c *Non-functions*: How should the tool represent everything in the formula that isn't a function: numbers, string literals, errors, references, etc? Since these don't accept arguments, they will adorn the tree as leaves, and their precise content won't affect the functions around them as long as their types are known. As such, in the visualization, all of these nodes are replaced and aggregated under their types.

3a *Suitable examples*: Goal 3 supports the inclusion of examples in the visualization as a way of tying pattern to example, but how should examples be chosen? For this, the simpler is the better, and we made the broad working assumption that shorter (by character length) functions are simpler, and thus chose the shortest available.

3b *Spreadsheet connections*: Is it possible to contextualize these examples even better? Yes, by leading the user directly back to the originating spreadsheet. The tool hyperlinks each example, then, back to the file that contains it, providing sheet, row, and column numbers if it doesn't open directly there.

D. Implementation

We can view the final visualization as the product of two discrete processes:

- *Collection*: Given a set of Excel sheets, the tool, written mostly in Java, uses Apache POI[5] to identify and iterate over every cell containing a valid formula. Afterward, it calls POI's formula parser to break the formula text into an ordered set of individual tokens, which the tool then parses into the tree-like form by which it is recorded. When all formulas have been analyzed like this, it produces JSON files for each top-level function in the set.
- *Presentation*: The JSON files, meanwhile, feed into the presentation code, implemented in Javascript with much help from the visualization library D3[6].

When the presentation code displays the tree, it shows two types of nodes with different meanings: the circles represent a function that has been observed at that structural place, and hovering over these yields a tooltip with supplementary information and examples; and squares, which represent the argument positions for the function which is their parent. For example, if the IF function has been observed with three arguments, 3 squares will extend from it as children. Clicking on the square labeled '1' will yield more circles, all of which are functions or values that the tool observed as the first argument in an IF function; clicking on '2' will yield those possibilities in the second position; and so on.

E. Limitations

Because the data collection depends on POI's formula parser, it affords no leeway or partial information from a formula; it either processes it perfectly or throws it out. As such, the visualization inhibits insight into anything with syntactical errors or third-party functions that can't be evaluated without additional tools. Note, however, that this does not include standard Excel errors like #REF! and #DIV/0, which the parser handles well.

Many functions have an expected order of arguments, but some, like SUM and MATCH, can be reordered in various ways and maintain their value. However, the representation that this tool uses reinforces for all functions that the order is important, even in these exceptions.

---

[4]https://support.office.com/en-us/article/Excel-functions-by-category-5f91f4e9-7b42-46d2-9bd1-63f26a86c0eb

[5]https://poi.apache.org/
[6]https://d3js.org/

A larger and related problem is the loss of argument combination. In the example in section B, we saw how the tools conveys which arguments are used most often on each side of an equals sign, but it does not encode how these whether references were actually compared to strings and numbers the most or whether it corresponded more to all the arguments beneath those two

IV. CASE STUDY

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus ultricies volutpat laoreet. Suspendisse tempus sapien metus, eget eleifend risus vehicula nec. Nulla facilisi. Aliquam non sapien ut tellus sagittis varius vel non nisl. Ut interdum nibh interdum odio ullamcorper, ac rhoncus leo tincidunt. Suspendisse potenti. Sed at scelerisque urna. Duis vehicula, nulla sed luctus aliquet, ante ante finibus arcu, at condimentum ante velit accumsan ligula. Sed suscipit, risus et blandit vestibulum, nibh felis mollis tellus, id pharetra nisi tellus sed augue. Cras nunc orci, semper auctor cursus sed, sodales eu risus. Etiam nec diam sed mi fringilla luctus sit amet quis neque.

Etiam porttitor lectus a augue volutpat suscipit. Pellentesque sodales dictum convallis. Cras varius pharetra tempor. Sed id finibus libero. Nullam aliquet posuere pellentesque. Sed rhoncus consequat lectus sed fermentum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Fusce euismod justo non velit molestie, vitae efficitur dui molestie. Ut molestie eros ullamcorper, convallis nibh vel, varius libero. Nulla sit amet tincidunt risus, sed porta nisi. Phasellus faucibus ut nisl vitae convallis.

Nulla efficitur diam diam, ac aliquam sapien vestibulum nec. Aliquam eu tempus sem, placerat dapibus metus. Sed augue tortor, hendrerit vitae ex et, vestibulum placerat erat. Donec rhoncus scelerisque nunc, sed elementum ipsum pharetra scelerisque. In bibendum massa eros, tristique sodales justo placerat at. Praesent egestas convallis nunc, in tempor ex tempus at. Suspendisse potenti. Maecenas mi tortor, aliquet a nibh nec, pretium dignissim nunc. Nullam eleifend volutpat facilisis. In condimentum ut ante nec ullamcorper. Sed finibus cursus justo vel pharetra. Praesent elit odio, semper eget dolor eu, tempor tristique eros. Praesent dapibus nec velit at aliquet.

Sed a tempor velit. Maecenas congue consectetur elementum. Vestibulum dolor ante, aliquet et augue sed, congue vestibulum ligula. Vivamus viverra ex quis justo varius sodales. Nulla finibus risus vitae orci porta, a ornare ex aliquet. Sed mi sapien, sollicitudin tempus justo vitae, eleifend auctor sem. Aliquam aliquam purus quam.

In rutrum tortor et enim fringilla, in dictum dui vulputate. Maecenas ut ligula et ipsum euismod ornare ut vel eros. Duis sit amet urna in tortor pretium convallis. Morbi vitae sollicitudin nisl. Etiam semper urna a ex aliquam malesuada. Aenean non bibendum velit, eu auctor metus. Integer vestibulum id turpis eget mattis. Donec diam nisi, rhoncus bibendum sollicitudin sed, eleifend vel massa. Curabitur ut ornare felis. Nullam viverra nulla in suscipit facilisis. Quisque mollis turpis id massa mattis sagittis. Ut vitae arcu a lectus gravida varius. Ut porttitor auctor rhoncus.

V. CONCLUSION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce tristique lobortis euismod. Suspendisse dapibus ultrices bibendum. Etiam accumsan augue metus, sit amet accumsan dui tempus eu. Donec sed nibh id tellus posuere luctus. Ut vel sagittis tellus. Phasellus dapibus pharetra nibh a consectetur. Phasellus dignissim nibh posuere, auctor mauris eu, vestibulum nulla. Aliquam mi nibh, maximus vel interdum non, molestie vitae ante. Donec vitae lacinia purus, eget maximus leo. Duis volutpat congue ligula non hendrerit.

Donec ac nibh imperdiet, hendrerit mi ac, viverra dolor. Vivamus tincidunt vehicula aliquet. In mollis egestas justo vel tincidunt. Donec in efficitur nisl. Donec eu nisl eu sem mattis malesuada et vitae lorem. Morbi urna purus, consectetur eget enim eget, interdum tincidunt diam. Etiam ultrices, metus nec fermentum porta, eros augue faucibus mauris, vitae ultricies nulla urna eget eros. Ut egestas, lorem ac volutpat pretium, est sem iaculis augue, et congue nisl massa vitae ipsum. Nulla facilisi. Donec metus nibh, laoreet et finibus ut, pretium eu ante. Fusce mauris enim, interdum nec consequat ac, interdum vitae ipsum. Mauris malesuada vitae arcu eu sagittis. Nullam euismod purus arcu, vitae consequat erat efficitur eu. Donec egestas felis sed mauris sagittis, at sodales neque lobortis.

Curabitur sed ultricies nisi. Sed pharetra non dui nec dapibus. Sed tincidunt lacinia mi at elementum. Suspendisse egestas risus scelerisque viverra vestibulum. In hac habitasse platea dictumst. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Donec fermentum libero eget libero iaculis porttitor. Curabitur libero nisl, varius ac lorem vel, ornare viverra erat. Nunc id magna est. Integer interdum tristique nulla, nec mattis nisl hendrerit a. Quisque turpis nisl, feugiat in sagittis a, viverra at massa.

-extensible to other spreadsheets?

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.