Department: Graphically Speaking
Editor: Rahul Basole

Notebooks for Data Analysis and Visualization: Moving Beyond the Data

Robert Kosara
Observable

***Abstract*—Notebooks are a relatively new way of analyzing data and creating visualizations. They differ from the common graphical user interfaces used for visualization tools in many ways, and have their own strengths and weaknesses. In particular, they allow easy sharing, experimentation, and collaboration, and provide context about the data for different kinds of users. They also integrate modeling, forecasting, and complex analyses directly with the visualization. We believe that notebooks provide a unique and fundamentally new way of working with and understanding data. By laying out their unique properties, we hope to inspire both researchers and practitioners to investigate their many uses, explore their pros and cons, and share their findings.**

BOTH WORK AND LEISURE have increasingly moved online over the last decade. We buy many things online instead of heading to a physical brick-and-mortar store, we stream music and movies instead of buying them on shiny disks, we share files and photos using cloud services instead of physical media, and similarly use collaboration features in Google Docs, Overleaf, or Figma instead of emailing drafts back and forth.

This is in stark contrast to the old model of creating standalone applications or perhaps running local, "on-premises" servers to host network applications. Cloud services naturally support sharing and collaboration, two crucial components of data work whose importance has been underappreciated in visualization research so far.

While computational notebooks have been around for a few decades, taking them online has made them much more powerful than before. The first notebooks were strongly focused on mathematics, like MATLAB and Mathematica, which both started in the 1980s. They were joined in the early 2000s by iPython, R markdown, Jupyter, and others. Notebooks are popular in data science and statistics because of their ability to integrate data access and processing, graphics, and text in the same place. They also provide an easy way to trace the origin and flow of data through all the processing steps.

Cloud-based notebooks naturally extend these capabilities with the ease of working in the browser, as well as the possibility of seamless

 Published by the IEEE Computer Society

real-time collaboration. Collaboration in visualization tools is still in its infancy, but is well established in word processors, such as Word and Google Docs. Notebooks being text-based can square the circle here by combining familiar text-based collaboration with the power and expressiveness of arbitrary code, visualization, and interactive elements (Figure 1).

LITERATE VISUALIZATION

Modern visualization and *business intelligence* (BI) products, such as Tableau and PowerBI, are built with graphical user interfaces (GUIs). The reasons are almost self-evident: ease of use, immediate feedback, lowering the barrier of entry for new users unfamiliar with command lines and code, etc. There is also a recent push for "low-code/no-code" tools, which are perceived as more approachable.

For all their advantages, GUIs do have their drawbacks. A GUI application has to be designed with a clear idea of the end user, which the developers and program managers might not have, or which might be too narrow or simply change over time. Applications often respond to this with tacked-on features that don't fully interact with the existing ones and can be difficult to use. A given visualization or dashboard can also be difficult to reverse-engineer or even fully understand. The myriad choices made in preparing, transforming, and filtering the data, the way different views interact, the choice of views, etc., cannot be easily traced back or documented.

Donald Knuth coined the term *literate programming* in a 1984 paper [6] to mean a mix of code and documentation in the same document. This was based on his TEX typesetting system, which lends itself quite naturally to mixing code and markup for documentation (including mathematical formulas). The same source document could be fed to two different programs that would either extract the program source code to be fed to a compiler, or run TEX to produce documentation together with beautifully typeset code. While this idea in its original form did not catch on, it certainly has echos in documentation systems like JavaDoc, JSDoc, TypeScript type annotations, etc.

Wood et al. applied this idea to visualization in their *literate visualization* work [8], by building a system that would integrate text and visualization code to create narratives. They cited Observable (which was in beta at the time) as one implementation of a similar idea.

Today's notebooks, like Observable and others, go well beyond Knuth's original idea by allowing the user to immediately run the code. This way, not only are code and documentation tightly integrated, but the results the code produces are right there as well. And beyond simple code execution, many notebook environments have adapted a reactive code execution model. This means that results are not static but are recomputed on demand when the user interacts with a visualization or model, or when the code changes.

For the notebook or dashboard consumer, this interactivity should extend beyond the ability to move sliders or change options for a model. A reader can edit a notebook (with complete history to be able to undo any changes) or create their own version of a notebook by forking it, testing out a number of ideas, and suggesting those changes back to the original author. This blurs the line between author and reader, and allows everybody to contribute without the friction of most other approaches.

SIDEBAR: Notebooks

According to Observable co-founder Mike Bostock, a notebook is "*an interactive, editable document defined by code. It's a computer program, but one that's designed to be easier to read and write by humans.*" [2]

In contrast to GUI applications, notebooks are based on text. A notebook consists of the text and code written by its author (or authors) as well as the results of that code. This can include computations of any complexity, database or API access, graphics including visualizations, as well as interactive and animated elements.

The initial idea of notebooks goes back to Donald Kuth's *literate programming* idea of mixing documentation and code, and some notebook formats are meant to be compiled into a static output presentation format. Modern notebooks, especially ones living in the browser, forego that step in favor of immediately (and often reactively) executing code to always show the results.

This allows not only a much more direct

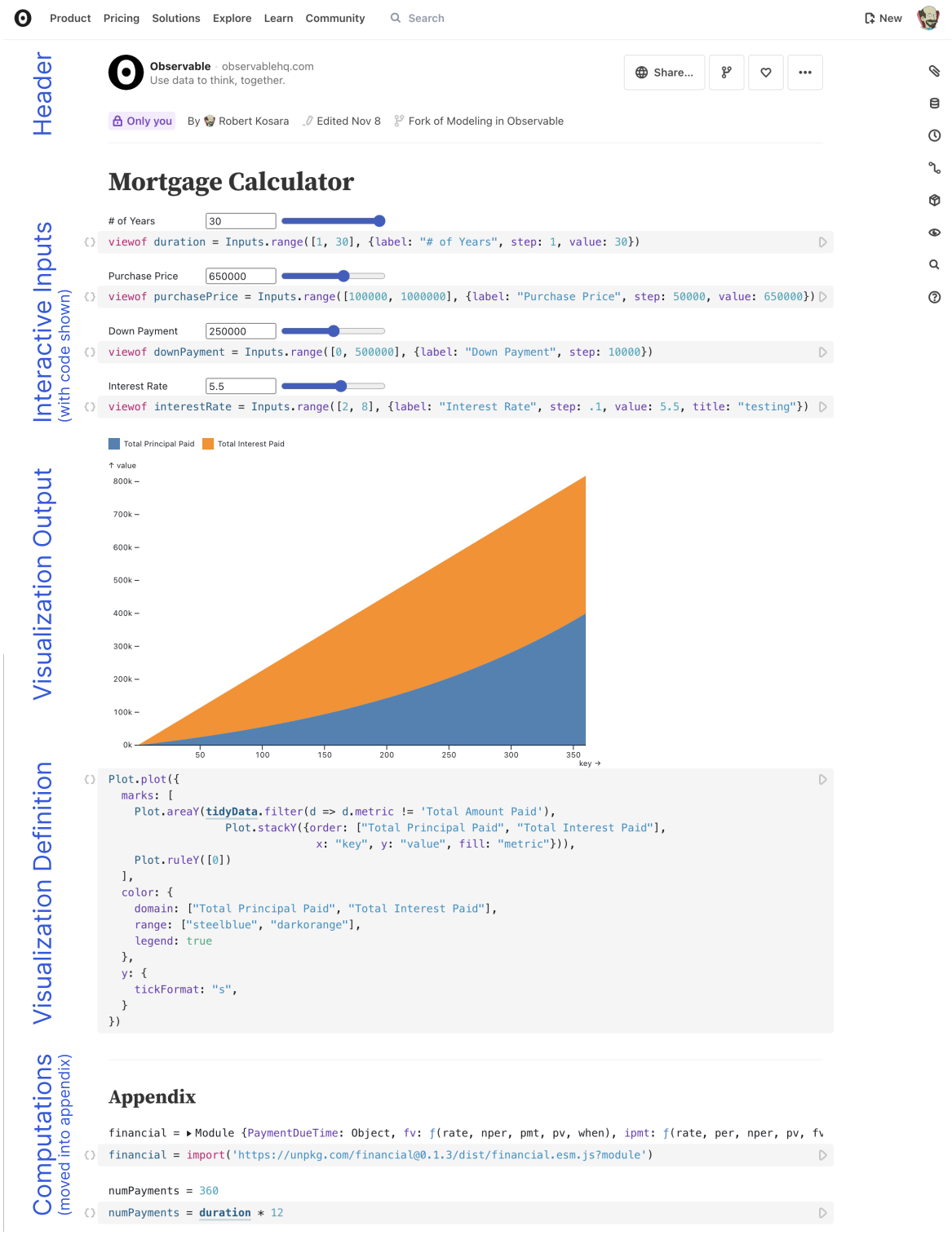

Figure 1. Notebooks contain code that runs in the browser. This can include data processing, visualizations, modeling, and interactive elements. This particular example implements a simple mortgage calculator that shows how principal and interest are paid off over time.

way of writing code, but also the inclusion of interactive elements that a reader of the document can manipulate. These changes then trigger the computations that depend on them to re-run and to show their outputs as numbers, text, or graphs.

COLLABORATION AROUND DATA

Visualization research seems stuck in a world of standalone visualization applications. The visualization world of yore was concerned with individual representations of data that users would interact with, but we've had powerful tools for visualizing data for decades now. Many of the problems users run into today are not which chart type to use or how to make it, but questions around the visualization itself: Where did this data come from? When was it last updated? Was any of it filtered or transformed? In what ways? How can the visualization, report, or dashboard be shared with different people in different roles, with different questions and needs? How can subject-matter experts and analysts collaborate with each other, as well as non-technical users such as managers and the decision makers who ultimately need the data to drive the business?

In a study of people presenting data in organizations, we found many different kinds of collaborations like this, in many different configurations of group size and composition, in many different organizations [3]. We only scratched the surface with our small study, and anecdotally know of many more scenarios. Similarly, a recent study of data scientists found intense communication and collaboration at each state of their work [7], and an earlier study of a complex science collaboration revealed similar patterns in the sciences [4].

This is the reality of data and visualization work in the real world today. Virtually none of it happens in isolation, and it increasingly happens in environments that naturally include collaboration. Just like documents are edited collaboratively today as a matter of course, data analysis and visualization require easy collaboration, co-authoring, and communication to be of value.

Notebooks lend themselves to these types of collaboration, similar to text documents, in ways that GUI-based (and standalone) visualization tools do not. The same kind of commenting that we are now used to from the likes of Word and Google Docs is possible in a notebook. Users without the technical abilities to work with the data themselves, or without the time to do so, can leave comments to ask questions or make suggestions. This is crucial for communication between different types of users with differing levels of technical expertise and job roles. While analysts might enjoy working together in real time in the same notebook (Figure 2), a higher-level manager or other collaborator might want to ask questions or provide additional information that can be incorporated without doing it themselves.

Collaboration can happen on another level, too: that of code reuse. Notebooks can easily include parts of other notebooks to reuse components, data, visualization definitions, etc. This allows an organization to build up a library of components and datasets, as well as use publicly available code, in ways that is virtually impossible in GUI tools.

Observable uses a forking model for its notebooks that is based on that of distributed version control systems such as git. Users can fork a notebook, which creates a copy that is connected to the parent notebook it was forked from. Changes made to the forked notebook can be sent back to the author of the parent as suggestions, which they can then choose to accept or ignore. While primarily a collaboration feature, forking is also useful when modifying one's own notebooks to quickly try out variations that can then be folded into the original or discarded.

In addition to modeling for the purpose of forecasting and planning, notebooks' ability to include arbitrary code also simplifies otherwise complex types of queries. Market basket analyses (which items are bought together) and cohort analyses (how do customers grouped by some criteria differ from others) in particular often require complicated and non-intuitive operations in BI tools. Implementing them in code may be simpler, allows for much more flexibility and control, and can be implemented once and reused many times —whereas in many GUI applications, users may need to follow a set of inscrutable steps every time they want to create a particular type of analysis.

Of course, this assumes familiarity with programming and at least its basic concepts. For people with a data science or analysis background,

Figure 2. Several people can edit a notebook at the same time on Observable, collaborating across different roles, departments, etc. Users who can't or don't want to modify the document can still leave and respond to comments.

this might be relatively straightforward. The step from using spreadsheets is larger, even though reactive programming can make that it easier for them. To the best of our knowledge, this has never been studied, despite the current interest in data science and the prevalence of data work in all kinds of business environments today.

BEYOND THE DATA

Data visualization by its very nature can only show the data that is there. This may seem obvious, but it limits the usefulness of visualization work for many uses that require more than just the data that is available.

This includes modeling and forecasting, what-if analyses, and questions that specifically look for gaps or new areas to explore. For example, a retailer looking to open a new store will not look at the data about its existing stores, but where there currently aren't stores. This requires the combination of many external data sources (on demographics, income, traffic patterns, etc.) together with a way to find the gaps in the current coverage.

Similarly, financial planning is the foundation for many important (and potentially very expensive) decisions, but it is based on projections, forecasts, and assumptions rather than data in the conventional sense. Today, this is commonly done by running models in one system or language, such as R, Python, or Excel, to generate data that is then exported to be used in a visualization or BI tool. Many questions are difficult to answer this way, however, and if the data for a particular scenario wasn't prepared in advance, a decision may have to be put off until that model can be run and the data imported.

Because notebooks allow the seamless integration of data access and arbitrary code, interactive models can easily be combined with data. In a notebook, the user can interact with the model directly and see its results shown immediately to compare against the data. Direct manipulation of the model allows for much faster and deeper exploration than having to generate data for a limited number of parameter combinations.

Modeling may involve complex operations, but it can also consist of simple, rapid prototyping of a quick idea that may be explored and discarded if it turns out to not be of value. This is similar to one of the strengths of spreadsheets [1] that allow for quick side calculations to double-check a number or test a different way of computing something.

Even when the focus is entirely on data, questions about the data are common in business and other contexts. Analyses and presentations typically aren't one-offs, but repeat every week, month, quarter, or year [3]. When looking at a visualization or analysis, it is important to know whether it is live or frozen at a particular time (there are use cases for both), when it was last updated, how it was processed, what other data it was combined with, when that reference data was last updated, where it came from, etc.

Notebooks allow the easy inclusion of large amounts of references, explanations, metadata, and links to further documentation. Key information can be shown with the data, with further explanations moved into an appendix or other notebooks. In contrast to a wiki or other documentation system that is detached from the data, documentation notebooks can reference the same data sources and thus integrate much more directly, and stay up to date, with the actual data.

Finally, notebooks by their very nature support building narratives around the data and analyses. They can include any amount of text to add context to the analysis and lead the reader through the different steps. Their linear nature also encourages reading from the top, which allows the author to introduce the subject matter,

frame the analysis, etc. Different scenarios can also be explored and compared easily, with a thread through them to explain why and what the reader can learn from each. And because notebooks are interactive, audience members such as decision makers can explore parameters and add their own twists and constraints. This allows a much richer back-and-forth between the different stakeholders than is currently possible.

MAKING DECISIONS

What is often overlooked in research is the ultimate reason for data analysis and visualization: to support decision making. These can be business decisions such as investments or the opening or closing of retail locations, they can be policy decisions that may impact millions of people, or they may be small but crucial decisions such as the choice of treatment for a single patient.

Virtually all decisions are made through processes that involve different people in different roles, bringing together their expertise, perspectives, and interests. The current separation between tools for modeling, data processing, visualization, presentation, and discussion creates friction and delays.

Early BI tools required the creation of import scripts and data extracts, and a cadre of experts who were able (and authorized) to modify dashboards following formal change requests. This created friction and delays in ways that are similar today's fragmentation of tools.

Notebooks can act as common spaces for sense-making, collaboration, and storytelling. By integrating the data processing pipeline with analysis, modeling, and visualization, they act like integrated development environments – only in a more literal, collaborative sense. Being able to trace steps and see the data, models, questions, and scenario explorations that led to a particular decision in one place makes the process more transparent and inclusive. Different roles at many different levels can be involved directly, rather than being kept at arms length and depending on a game of post to receive enough information.

OPPORTUNITIES

What can notebooks do for data analysis and visualization? What are their strengths and weaknesses? This new paradigm presents many interesting topics for conducting novel research and for pushing the envelope of the practice of data science and visualization.

We've outlined some of their strengths and weaknesses here, but there is scant research on this topic so far. This is especially true because it involves not just single data visualizations or a single user, but multiple people working together across data analysis, visualization, and decision making. How do we even study such a complex topic and do the real-world scenarios justice? This is challenging work, but at the same time very valuable for the work that needs to be done in the real world today.

In addition to the research questions, how does the practice of data visualization change with notebooks? Does this idea of literate programming hold up? Can people work in code rather than GUIs? Or are notebooks too scary for non-technical users?

Where are notebooks going? The basic idea is easy enough to understand, but what needs to be built on top of the current generation? Do we end up building GUIs on top of them? Are notebooks destined to end up converging towards some kind of dashboard model that is so prominent in Business Intelligence software? And if so, can they maintain their differences?

The ability to mix data processing, visualization, and modeling together with the natural collaboration that notebooks not just enable, but invite, blurs the lines between author and reader. How does this change the way people think about their data and their work? Will they end up being more engaged as a result? Anecdotally, we know that the vast majority of dashboard users never use any interactive features. Can the more text-based approach of notebooks change that?

Notebooks are also increasingly used in educational settings, and not just to teach data visualization. We haven't even begun to consider the possibilities (and, potentially, challenges) here, with students being able to interact directly in a document, instructors being able to observe and help, etc.

CONCLUSION

For data to be the point where analysts, subject-matter experts, and managers come to-

gether, they all need meaningful ways of answering their specific questions and contributing to the work. This may mean tweaking a model or filtering data, but it may also mean asking tough questions.

Notebooks present a new paradigm for data analysis that is different from the tried-and-true standalone GUI application. They open up many new possibilities for work in a more modern, collaborative, and integrated way – and at the same time pose many interesting new questions about the role of visualization as part of the work with data.

A better understanding of notebooks' strengths and weaknesses will allow us to push their abilities further, and with them the state of the art of visualization research and practice.

REFERENCES

1. Bartram, L., Correll, M., Tory, M., Untidy Data: The Unreasonable Effectiveness of Tables, *Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 686–696,, 2022.
2. Bostock, M., Introduction to Notebooks, https://observablehq.com/@observablehq/introduction-to-notebooks, 2017
3. Brehmer, M., Kosara, R., From Jam Session to Recital: Synchronous Communication and Collaboration Around Data in Organizations, *Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 1139–1149, 2022.
4. Borgman, C. L. , Wallis, J. C., Mayernik, M. S., Who's got the data? interdependencies in science and technology collaborations. *Computer Supported Cooperative Work (CSCW)*, vol. 21, no. 6, pp.485–523, 2012.
5. Dimara, E., Zhang, H., Tory, M., Franconeri, S., The Unmet Data Visualization Needs of Decision Makers within Organizations, *Transactions on Visualization and Computer Graphics*, 2021.
6. D. E. Knuth. Literate programming. *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984.
7. Pang, Y., Wang, R., Nelson, J., Battle, L., How Do Data Scientists Communicate Intermediate Results? *Symposium on Visualization in Data Science (VDS)*, 2022.
8. Wood, J., Kachkaev, A., Dykes, J. Design Exposition with Literate Visualization. *Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 759–768, 2019.

Robert Kosara is with Observable, Inc. He received his M.S. and Ph.D. degrees from Vienna University of Technology. Robert's research interests include visualization encodings, perceptual issues in visualization, and the broader area of communication and storytelling with visualization. Contact him at rkosara@observablehq.com.