Journal Title
XX(X):1–16
©The Author(s) 2023
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Lodestar: Supporting Rapid Prototyping of Data Science Workflows Through Data-Driven Analysis Recommendations

Deepthi Raghunandan[1], Zhe Cui[2], Kartik Krishnan[1], Segen Tirfe[1], Shenzhi Shi[1], Tejaswi Darshan Shrestha[1], Leilani Battle[3], and Niklas Elmqvist[1]

Abstract
Keeping abreast of current trends, technologies, and best practices in visualization and data analysis is becoming increasingly difficult, especially for fledgling data scientists. In this paper, we propose LODESTAR, an interactive computational notebook that allows users to quickly explore and construct new data science workflows by selecting from a list of automated analysis recommendations. We derive our recommendations from directed graphs of known analysis states, with two input sources: one manually curated from online data science tutorials, and another extracted through semi-automatic analysis of a corpus of over 6,000 Jupyter notebooks. We validated Lodestar through three separate user studies: first a formative evaluation involving novices learning data science using the tool. We used the feedback from this study to improve the tool. This was followed by a summative study involving both new and returning participants from the formative evaluation to test the efficacy of our improvements. We also engaged professional data scientists in an expert review assessing the utility of the different recommendations. Overall, our results suggest that both novice and professional users find Lodestar useful for rapidly creating data science workflows.

Keywords
Computational notebook, visualization recommendation, Markov chain, data science, Python.

Introduction

Data science is still a nascent and emerging discipline, which makes it challenging for analysts to learn and keep up with new tools and techniques. There is already a dizzying array of libraries, such as Scikit-Learn, Pandas, and TensorFlow, and best practices and workflows change often. Furthermore, few standardized methods exist for data analysis: many times, the exact data transformations, computations, and analyses needed depends on the data, task, and user. This means that cookbook methods or simple templates are insufficient to teach fledgling analysts how to tackle realistic and ever-changing data science problems.[27]

We present LODESTAR, an interactive and visual sandbox for independent learning of analysis methods and best practices in data science. Our aim in developing Lodestar is to simplify the process of finding and experimenting with new methods by providing automated, data-driven recommendations. The vision is for Lodestar to be a self-contained environment for rapid learning and prototyping that combines everything the user needs to infer the function and purpose of an analysis step in one place.

The Lodestar system shows a sequence of analysis steps in the form of Python code cells (see Figure 1), like in a computational notebook interface (such as Jupyter notebook[25]), but enables the user to initially select from and interact with self-contained code cells without having to write any code. The user merely selects which data-frame to analyze, and the system displays a ranked list of recommendations of analysis steps to be executed on that data. Each analysis step is represented by an interactive visualization in the notebook interface, giving the user insights into its output and behavior. Furthermore, users can view the corresponding code for any analysis step, and even export the resulting notebook from Lodestar, providing flexibility in how users learn from and interact with Lodestar's analysis recommendations.

Lodestar provides recommendations for the user's next analysis step based on the current state of the analysis and the dataset being analyzed. Recommended analysis steps and workflows are derived from two sources representing current best practices in data science: (1) existing data science tutorials from online academies and training materials (i.e., *expert recommendations*), and (2) common analysis patterns mined from a large corpus of publicly available Jupyter notebooks[42] (i.e., *crowd recommendations*). The code cells extracted from each source are manually curated, then programmatically clustered into synonymous analysis steps, and inserted into a large directed graph of connected cells representing common analysis workflows. The Lodestar recommendation engine can then identify and rank the most relevant analysis steps given the user's current position in the graph.

We developed Lodestar using an iterative design process through three separate user studies. In the first study, we used early feedback from six novice data scientists to

[1]University of Maryland, College Park, MD
[2]Google, Inc., Mountain View, CA, USA
[3]University of Washington, Seattle, WA
Email: {draghun1, kkrishn1, segent, sshi1234, elm}@umd.edu, shreshta.tj@gmail.com, zhecui@google.com, leibatt@cs.washington.edu

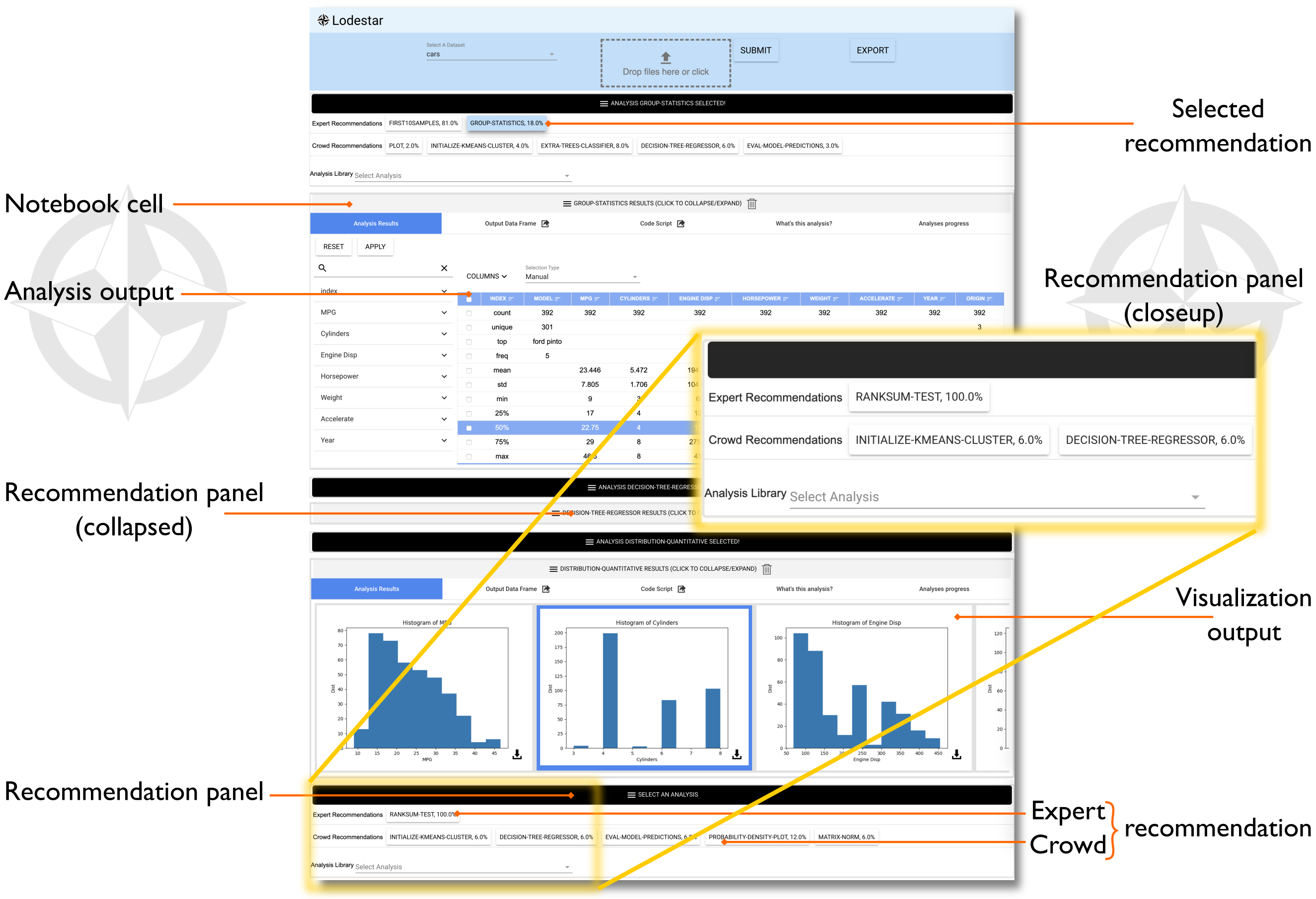


Figure 1. Lodestar web interface. The top panel, above the selected recommendations, provides a data selection menu. The black dividers between sections are recommendation panels combining suggested analysis steps from various sources (called advisors). Areas that have graphs and analysis outputted are analysis cells, each with multiple tabs: “Analysis Results” gives charts or tables, “Output Dataframe” and “Code Script” shows the outputs and current code block, and “What’s this analysis?” gives a brief description of the analyses.

improve the interface design in a formative user study. Once we made improvements to the interface design, we asked three returning participants and seven new participants to evaluate the improved version of the interface in a summative user study. Our findings show that key Lodestar interactive features, such as automated recommendations, a visualization of the full analysis workflow, a code review pane for suggested analysis steps, and support for Jupyter Notebooks, provide significant value to those who are learning data science. Finally, in a third study, we invited three professional data scientists to evaluate Lodestar recommendations and its recommendation engine. This evaluation showed that Lodestar recommendations provide an easy way to explore data and analysis techniques.

In this work, we make the following contributions: (1) a recommendation system involving two sources of data analysis practice: crowd-based and expert-based; (2) a sandbox interface design integrating visualizations, interactions, and code to facilitate learning about new data analysis techniques; (3) results from formative study and a summative study evaluating the Lodestar interface; (4) results from a study evaluating Lodestar recommendations and recommender engine; and (5) a novel data analysis architecture that integrates a recommender system with an interactive interface. All our materials have been made available on OSF: https://osf.io/pztva/

Motivating Scenario

Chewie is a recently retired journalist, and is hoping to devote his retirement to his true passion: traveling the world. He owns a condo in a nice downtown area, and is now trying to decide whether selling or subletting his house is the better choice given the changing market. What will yield him the most funds to travel in the long run? Unfortunately, while Chewie has long experience finding the facts, he has no training in data science or temporal forecasting. He turns to Lodestar for help.

Chewie talks to his realtor, who is able to quickly get him a dataset of recently sold homes in the neighborhood. The resulting CSV file contains approximately 500 homes sold in the last five years, including the price, address, square footage, date sold, the income of the house owners, and school statistics. The realtor also provides him with another dataset of 1,000 rentals in the city for the last 10 years, including monthly rent, address, size, and year.

Chewie first creates a Lodestar notebook and loads the 500 sold houses. The first few recommendations include several descriptive statistics that give an overview of the dataset. He chooses `scatterplot-regression`, which renders scatterplots between various attributes with a estimated regression line. He then selects the shuffle-split analysis from the expert advisor, which is described as the first step of training a model. The human-readable description informs him that this is a standard method to randomly select a holdout dataset for training, and one for testing. Then he goes along with the next top recommendation fit-decision-tree, to actually train a decision tree model; this shows up from both the expert and crowd advisors, as it is a common step for forecasting. He is now ready to make predictions and uses the Lodestar default parameters to input the data for his house: selling it now, versus in five years. The model suggests that selling his house in five years instead of now will net him an extra $25,000.

Satisfied, he creates a new Lodestar notebook and repeats the process for the rental dataset. Again, the parameters of his house will yield him a suggested rent—$1,200 per month—as well as its market increase over the five years. He takes his figure and manually deducts his monthly mortgage payment, and realizes that his rental income over those five years will be close to $5,000 more than selling the condo now. His decision made, Chewie contacts his realtor and tells him to list the condo for rent, and then turns to planning his escape to warmer clime.

Background

Lodestar was architected to encourage best practices in sensemaking. Richard Hamming described sensemaking as "the process of searching for a representation and encoding data in that representation to answer task-specific questions".[43] Dubbed the *sensemaking loop*,[36] each sensemaking iteration works to refine and build on the previous insights—ultimately enabling the analyst to address less specialized audiences.

In combination, these iterations make up the data science workflow. Analysts often use visualizations or other types of intermediate results to guide further analysis. However, these results can sometimes be dead ends. Kandel et al.[22] found that analysts will commonly overcome dead ends by backtracking and exploring new branches.

Interactive Visualization Design Environments

Many visualization systems and toolkits are designed around specific data analysis tasks, making the analysis process easier to perform. Excel supports basic visualization and data transformations. Shelf-based visualization environments such as Tableau (née Polaris[51]) allow easy configuration of visualizations through drag-and-drop of data attributes and metadata onto "shelves" representing visual channels. This approach is flexible enough for even novice users to construct a wide range of visualizations. Interactive visual design environments such as Lyra,[45] iVoLVER,[32] and iVisDesigner[39] utilize direct manipulation to allow users to bind data to visual representations. More recently, Data-Driven Guides,[24] Data Illustrator,[28] DataInk,[65] and Charticulator[40] provide advanced tools for representing data items as visual elements and mapping their attributes to data dimensions. Keshif,[66] a faceted visualization tool, generates grids of predefined charts to support visual exploration by novices. ExPlates[20] uses fluid drag-and-drop interaction to support spatialized data analysis.

Visualization development toolkits such as D3[7] and Protovis[6] provide fine-grained control over designing interactive visualizations, but require significant programming expertise to use. Visualization grammars, such as ggplot2,[61] Vega,[47] and Vega-Lite,[46] abstract away implementation details, but still require programming knowledge to use. Furthermore, even advanced visualization tools, toolkits, and grammars offer only limited functionality for manipulating the data, and only support a small number of statistical functions.

Visualization Recommendation

The purpose of visualization recommendation is to suggest relevant visualizations to the user to facilitate data analysis[18], where the visualizations are fully designed in advance and therefore directly accessible to the user. It was first proposed by Mackinlay[29] in 1986 with automatic design of effective presentations based on input data. The work combines expressiveness and effectiveness criteria from studies such as those by Bertin[5] and Cleveland et al.[8] to recommend appropriate visualizations. In 2007, Tableau's Show Me feature[30] revealed a commercial product with the implementation of these ideas. Following the idea of Mackinlay's automatic visualization, Roth et al.[41] enhances user-oriented design by completing and retrieving partial design graphics based on their appearance and data contents. The rank-by-feature framework[48] ranks histograms, scatterplots, and boxplots over 1D or 2D projections to find important features in multidimensional data. SeeDB[59] generates all possible visualizations given a query and identifies the interesting ones. Perry et al.[35] as well as van den Elzen and van Wijk[57] propose generating small multiple visualizations shown as thumbnails using summary statistics.

In the last few years, recommender systems have become widely used for visualization. Voyager[62] generates a large number of visualizations given a user-specified partial specification, and organizes them by data attributes. The generated visualizations are rendered as cards on a scrolling view. Saket et al.[44] propose the Visualization-by-Demonstration framework, which allows users to provide incremental changes to the visual representation. The system recommends potential transformations such as data mapping, axes, and view specification transformations. Zenvisage[49] automatically identifies and recommends desired visualizations from a large dataset. Voyager 2[63] extended the original Voyager through wildcard functionality that explores all possible combinations of attributes. Draco[33] automates visualization design itself using partial specifications and a database of design knowledge expressed as constraints. VizML learns what visualizations to recommend by training neural network models on millions of visualization designs made using Plotly.[19] Similarly, Qian et al.[37] uses learning-based approaches to generate relevant visualizations based on data. Most recently, and uniquely, Solas[14] learns to provide visualization recommendations using a user's analysis history.

Several tools extend these ideas to recommending analytical insights and data processing steps. "Top-K insights"[53] provides a theory for generating top K insights from multidimensional data. Similarly, Foresight[11] presents the top K insights in a dataset from 12 insight classes using a corresponding visualization. DataSite[10] organizes significant automatic findings in a specific feed of notifications. Finally, Voder[50] builds on a similar feed as DataSite to provide "interactive data facts" using visualizations.

Our proposed Lodestar system combines these ideas from visualization recommendation with an analytical perspective, and allows stringing together such analytical steps into a sequence. There are some existing efforts on recommending data analysis techniques and workflows. Yan et al.[67] demonstrate that online repositories of computational notebooks can be a valuable resource for modeling and testing a recommendation system for data cleaning techniques. Milo et al.[3] take this a step further by automatically generating entire data exploratory workflows using deep reinforcement learning techniques. Our system builds on these works by presenting a holistic model and code mining pipeline for deriving new recommendation features in a data-driven way, whether for data visualization, data preparation, or data analysis workflows. Essentially, Lodestar extends the idea of automated recommendations to the entire data science pipeline, rather than visualizations only.

Interactive Notebooks

Donald Knuth's notion of a "literate" form of programming,[26] which merges source code with natural language and multimedia, has extended to the concept of *literate computing* in the form of computational notebooks,[25] that combine executable code, its output, and media objects in a single document. This has proven to be very useful for rapid prototyping and exploration as well as for replicability and communication in data science.[42]

Because of their success, with adoption even at the level of entire organizations,[56] notebooks have enjoyed significant progress in recent years. The new generation of computational notebooks, such as Google Colaboratory and Codestrates,[38] enable synchronous collaboration. The JavaScript-based Observable notebook also supports reactive execution flows.

Visualization in particular has recently begun to adopt computational notebooks. Altair[58] builds on Vega[47] and Vega-Lite[46] to provide statistical visualizations in Python, and thus in Jupyter Notebooks as well. Idyll[9] supports a notebook-like markup language to create interactive data-driven document for communication. Vistrates[2] provides a collaborative visualization workflow in a notebook. Observable leverages computational notebooks to also provide a collaborative visualization platform. Literate visualization[64] integrates the visualization design process with the choices that led to the implementation.

End-user and live programming paradigms have proven useful in creating intuitive interactions with visualizations found in computational notebooks. For example, Wrex[12] and Mage[23] leverage user interactions on data visualizations to automatically generate exemplar code. Both tools demonstrate the link between code and visual interactions. Torii[17] uses a live programming model to enable easy maintenance and reuse of source code for building tutorials. These systems not only add to the number of ways users can interact with their literate document, but also connect code and visualization so as to facilitate iterative analysis.

Design Requirements: Formative Study

Our goal is to make Lodestar an interactive and visual sandbox environment for learning and experimenting with new data science methods in a data-driven way. We also wanted to make data science universally accessible to fledgling data analysts and enthusiasts alike. These core ideas helped us compile a set of design requirements and some preliminary prototypes. In this section, we outline our major design requirements, and report on a formative study conducted to validate and refine our approach to the Lodestar interface design and system development processes.

- **D1: Informed by best practices.** Recommendations should be drawn from current practice, empowering those new to data science to learn how to effectively analyze data.[27,30]

- **D2: Prioritize analysis steps over code.** Our intended users are trying to analyze data in a fast and fluid fashion, but may not yet be familiar with specific libraries or modules needed to complete different analysis steps. Lodestar needs to build a bridge between the high level analysis steps common in data science, and the low-level code needed to accomplish these steps.[31] For example, recommendations should be immediately relevant and situated within the overall data science pipeline to enable users to progress in their analysis.

- **D3: Enable independent exploration.** To ensure that users can explore their data independently, educational interface elements must also be incorporated to automatically provide documentation and clarification of system behavior.[13] Furthermore, intermediate and final results should be presented using visual representations that can be easily interpreted regardless of user expertise.[55]

We conducted a formative user study to evaluate the usability of an early prototype of the Lodestar system, which we used to validate our initial design requirements and refine the design. Questions posed to the participants in the formative study protocol can be found in the supplemental material.

While we refer to Lodestar as a computational notebook, we note that building a fully-fledged notebook system from scratch is a vast engineering effort. Our goal in this research project was to focus on novel aspects of recommendation for data science and visualization, so our resulting notebook lacks significant features normally associated with such systems.

Study Design

The study was conducted over a period of one month in which we interviewed 6 fledgling analysts and data scientists; all undergraduate university students. We focused on recruiting university students, since they are generally learning data science methods for the first time and thus could provide helpful insights in our design process. Each student had demonstrated knowledge of data science fundamentals through attending a university-level introductory data science course and/or other relevant machine learning/data science experience. Although not a prerequisite of the recruitment process, some students also had experience performing analysis on platforms such as Excel and Tableau.

Method

Each interview lasted for 60 minutes and was divided into three phases. Prior to the interview, each participant signed a consent form, allowing us to record audio and screen capture throughout the duration of the interview. The first phase consisted of questions, delivered verbally, that assessed the participant's recent experience in learning data science techniques and tools through classes, side projects, research, and other such activities.

The second phase of the interview was dedicated to introducing an early prototype of the Lodestar system in which participants were given a brief 2-minute description of Lodestar and associated goals. The next 5 minutes were spent giving the participant a cursory tutorial of the system. For each participant, the tutorial was given using a pre-written script and with the same sample dataset to give each of them equal knowledge of the system prior to their exploration. The participants then spent the next 15-20 minutes using the Lodestar system to conduct exploratory data analysis on a dataset of their choosing. We restricted their choices to two datasets; the Boston House dataset from a Udacity tutorial,[*] and the Cars dataset. During this exploratory session, participants verbalized their thought process, questions, and comments with a think-aloud protocol. We encouraged participants to "to use any and all features of the Lodestar system" and to "explore whatever aspects of the data [they found] interesting." Participants were allowed to end the session before the allotted time expired if they were satisfied with their results.

The third and final phase of the interview consisted of a post-exploration questionnaire that asked participants to describe the utility of Lodestar for their common data analysis tasks. Participants were specifically asked if they would adopt Lodestar to learn new data science techniques.

All sessions were held in a lab environment using Google Chrome on a Macbook Pro with a 15-inch Retina display. Audio was recorded using the built-in voice recording application on a mobile device. Screen capture was done using Apple's QuickTime Player. Observational notes from the study coordinators, text responses from our questionnaires, and audio and video recordings were collected for further analysis and prioritization of design requirements and functional features of the existing prototype. We used the the participant responses from the think-aloud and questionnaire portion of the study to illustrate the themes with respect to the Lodestar enhancements.

Results

Our formative study found that a majority of participants were in favor of using Lodestar in their daily work, but suggested several modifications to make the system more useful. For the sake of brevity, we focus primarily on summarizing their constructive feedback below (participant IDs start with "FP"):

Provide Clear Documentation & Context: Our early prototypes did not include tooltips or descriptions of analysis steps. Several participants highlighted the need for increased transparency in the interface. Specifically, they wanted clearer naming conventions, documentation of features and methodologies (e.g., the difference between expert and crowd recommendations), and explanation of expected system behavior. For example, some participants had difficulties understanding the meaning of certain user interface elements. Participants asked questions such as "*what are these percentages?*" (FP6), or "*[what do] the columns on the left side represent?*" (FP5). Participants FP1, FP2, FP3, and FP5 also asked if there "*is actually a way to view the entire dataset?*" (FP2).

There were many questions specific to the meaning of recommendations. For example, FP3 said "*I think the names [are] misleading... there were some really complicated names for just a simple linear regression. [It] should just be changed [to more] obvious names.*" Similarly, FP2 suggested that there should be "*a longer description [...] [or] some way to show their effectiveness without the user having to Google search them.*" These misconceptions indicate that better documentation is needed to help new users understand the interface.

Improve Tracking of Analysis Progress: Several participants wanted to be able to see what phase of the data science process they were in based on the current state of their analysis workflow. Our early prototypes did not include the feature to track previously selected analysis. FP4 drew parallels with a restaurant order tracker, where Lodestar should partition each part of the data science process into separate steps, and group analysis recommendations into these steps. Users would then be able to better understand their progress within the data science process.

Enable More Granular Control: The early Lodestar prototype only allowed users to choose from pre-loaded datasets, and did not provide any export or customization functionality for analysis steps. However, multiple participants expressed the desire to import their own dataset and export their own code for later sharing and reuse. Participant FP4 said that they would be frustrated if they wanted to "*export it or make some changes in the data or [try] to do something that is not supported by Lodestar [while] not having any way of doing so.*" Participants also highlighted the need for more control over what parameters or attributes were being passed into different analysis steps, such as selecting specific attributes when generating visualizations or executing regression analyses. These observations suggest that users should be able to customize analysis steps and export their current analysis.

Further Refinement of Lodestar

Though participants could see promise in providing automated recommendations (design requirement **D1**), the expressed need for more tracking of workflow structure and progress also reinforces design requirement **D2**. Without additional context to help users situate themselves within the broader data science process, users can easily lose their train of thought, hindering their analytic flow. The need for

[*]The Udacity tutorial is available here: https://github.com/sajal2692/data-science-portfolio/blob/master/boston_housing/boston_housing.ipynb.

more documentation and control observed in our formative study supports design requirement **D3**. Without adequate information, users are unable to explore new data analysis techniques and interpret the results in Lodestar on their own. Users also find it difficult to tailor their explorations to their specific needs without access to the code.

These points of feedback served as motivation for additional iteration on the Lodestar feature design. Specific features that were added as a result of this study included the ability to export the user's notebook to an `.ipynb` file for use outside of the system, a visual tracker that displays the progress of the user's analysis in each output cell, showing which recommendations have been chosen so far, and descriptive tool-tips of the different analysis techniques in each output cell.

The Lodestar System

LODESTAR is a *data analysis recommender*, i.e., a system that interactively suggests the next step to take in an analysis workflow (D1). Lodestar is designed in the style of an interactive computational notebook, and generally inspired by the designs of existing notebooks such as Jupyter, Observable, and Google Colaboratory. Given Python's broad popularity in data science contexts,[42] we chose to focus on Python as our target environment.

System Overview

Lodestar consists of four main components, shown in Figure 2: a browser-based *notebook interface*, an *interactive computing protocol*, a *recommendation engine* to suggest analysis steps, and a server-side *kernel* to execute analysis steps. The protocol manages communication between the client and server (commands as well as computational results), and the kernel on the server side runs each analysis step that the user selects using an interpreter. The Flask server handles all of the client requests for data processing, analysis, and recommendations, with different endpoints.

Lodestar emphasizes an iterative workflow design where analysis steps are added progressively, one at a time, providing fine-grained control to the user (D3). To help users focus more on analysis steps and best practices (D1, D2) rather than low-level code, Lodestar allows the user to rapidly choose from a list of recommended analysis steps. These recommendations are displayed in the form of buttons, so a user can easily select and execute an analysis step of interest with a single click. Furthermore, these recommendations are mined from recent Python tutorials and active GitHub repositories of Jupyter Notebooks, enabling the user to construct new analysis workflows based on best practices in a data-driven way.

Notebook Interface

The Lodestar interface (shown in Figure 1) is an interactive notebook providing a literate computing environment[26] that runs in a web browser on the client. Similar to existing computational notebooks, the Lodestar notebook is a linear document that the user can selectively edit and execute. The interface contains three major components: a menu panel at the top, one or more notebook cells, and recommendation panels for each cell. The notebook cells and recommendation panels dynamically appear and update within the notebook interface in response to user interactions.

The user begins their analysis using the menu panel to load an existing dataset or a new dataset (in CSV format) into the system. Once a dataset has been loaded, Lodestar generates a recommendation panel within the notebook interface, providing the user with an initial set of recommended analysis steps. We refer to the actual code behind each analysis step as an *analysis block*, and the displayed result of executing the analysis step as a *notebook cell*. From this point onward, the analysis process forms a cycle that repeats until the user is satisfied with their new workflow:

1. The user **selects** an analysis step from a recommendation;
2. The kernel **executes the analysis block** on the server;
3. The notebook **displays output** by appending a new cell; and
4. The notebook **generates a new panel of recommendations**, based on the user's previous selection.

When the user is ready to migrate their workflow to a complementary tool, for example to iterate on the code directly within a code editor, they can export the Lodestar workflow as a Jupyter notebook file.

Recommendation Panel

Every notebook cell in the Lodestar interface has an accompanying recommendation panel, allowing the user to extend their latest analysis step by one cell. When the user selects an analysis step from a recommendation panel, a new notebook cell is generated for the selected recommendation, along with a new recommendation panel underneath. Lodestar uses the output of the preceding notebook cell as the input for executing any analysis step selected in this recommendation panel. Each panel provides two sets of recommendations, one from a *crowd advisor* and one from an *expert advisor*. The crowd advisor sources recommendations from online data analysis repositories such as GitHub. The expert advisor sources recommendations from educational resources such as textbooks, online classes or online tutorials. Crowd and expert advice are presented separately as a way to highlight a data point the user can take into consideration while choosing a recommendation. We believe that this type of transparency betters independent exploration (design requirement D3).

If a user is unsatisfied with a given set of recommendations, they can choose from Lodestar's full catalog of analysis steps in a drop-down menu at the bottom of each recommendation panel. This list is available in the supplementary materials.

Notebook Cell

Once a selection is made in a recommendation panel, the selected analysis step is highlighted and the results are displayed in a new notebook cell, allowing the user to review their past selections and the corresponding results with each subsequent step. Furthermore, the user is able to go back and update the results at any time by selecting a different analysis step in any of the previous recommendation panels. Any cell can also be deleted, which triggers the removal of all downstream cells that depend on the deleted cell. In this way, Lodestar maintains a linear structure in the notebook, making it easier for users to navigate within the analysis.

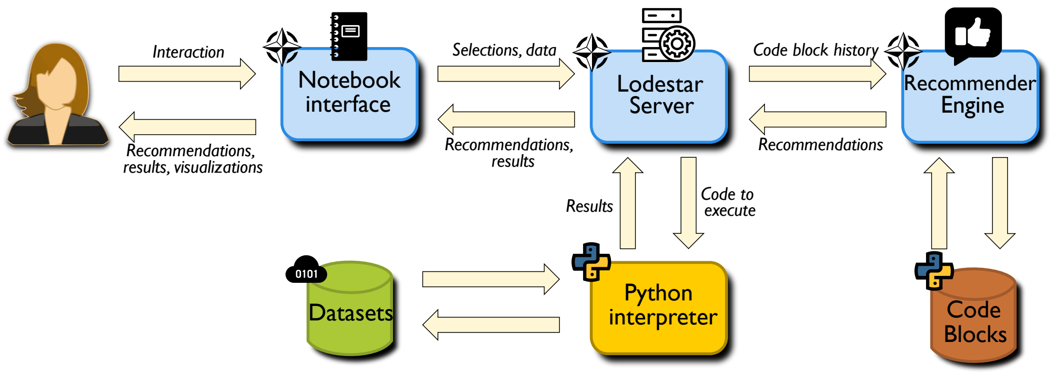


Figure 2. Overview of the Lodestar architecture. The user interacts with the notebook interface and selects either a data set to bootstrap the notebook or an analysis step within a guided workflow. The notebook interface sends the selection as a request to the Lodestar server. The Lodestar server sends requests to the recommendation engine for subsequent recommendations based on current selections (data or analysis). Lodestar server also sends a request to the Python interpreter to execute any selected analysis. Results from both these requests are sent back from the Lodestar server to the notebook interface for the user to view and interact.

To help users understand the functionality of each recommended analysis step and its purpose within the context of the larger data science process, notebook cells consist of five tabs. Each tab describes the behavior of the analysis block represented by this notebook cell. We refined the design of each tab based on the feedback we received from the formative study:

- **Output Data Frame:** Default view that renders the output data frame produced by executing the analysis step as a table.
- **Analysis Results:** Displays the raw results produced by the analysis step (e.g., `print` output or Seaborn visualization).
- **Script:** Displays the Python code within the analysis block.
- **"What's this Analysis?":** High-level description of the step.
- **Analysis Progress:** Displays the chain of analyses leading to the current analysis step, where each step has a name.

Exporting Code and Results

When the user is ready to migrate their analysis workflow to a related tool, they can export content directly from Lodestar. To export the code for a specific analysis step into an independent Jupyter notebook file, the user can click on the export button next to the *Code Script* tab of the corresponding cell. To export the entire analysis workflow, the user can click on the export button on the menu panel at the top of the interface. Similarly, Lodestar enables users to export the output data of any displayed notebook cell as a CSV file. To do this, the user clicks on the export button next to the *Output Data Frame* tab. The user can also download the visualizations displayed in any notebook cell as PNG files.

Advisors and Recommendations

The Lodestar recommendation engine is based on the notion of an *advisor*: a source of analysis recommendations. Lodestar supports multiple advisors, each consisting of a library of analysis steps and a set of advisor-recommended transitions between analysis steps (i.e., a recommendation graph). In our current implementation, we use two advisors: a "crowd" advisor drawn from our semi-automatic code analysis, and an "expert" advisor drawn from the manual code curation. For each advisor, the recommendation panel will show a list of up to five recommendations, ordered by probability, or how frequently this analysis came next in the recommendation graph.

In this section, we describe how we build our recommendation graphs for the expert and crowd advisors, and how we enable Lodestar to identify equivalent or related states across both graphs.

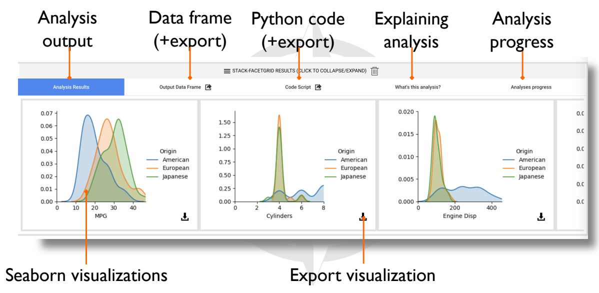


Figure 3. Figure grid. Visualizations generated by an analysis block using the Seaborn statistical data visualization package for Python.

Recommendation Graph

Lodestar models transitions between analysis steps by treating analysis workflows (e.g., existing tutorials or computational notebooks) as paths taken through a network graph. Each node in the graph is an analysis step, and a directed edge appears in the graph for each pair of consecutive analysis steps observed in a workflow. Lodestar leverages the relative frequency of these transitions to predict which analysis steps are likely to occur next. The particular graph structure used in Lodestar is a Markov chain, and the final computed graph we refer to as a *recommendation graph*. We believe this to be a good initial step towards modeling analytical decisions but, acknowledge that more complex factors influence their construction. We hope to explore this further in future work.

Lodestar traverses the recommendation graph one state at a time for each user input (i.e., choice of analysis step). As a result, our recommendation approach does not require maintaining specific state about the analysis itself. Instead, the location in the Markov chain serves as state, and transitions (e.g., recommendations) thus depend only on the current state. In this way, recommendations are agnostic of the data being analyzed, thus allowing users to draw from a wider range of tools and techniques (D3).

We can infer these recommendation graphs programmatically by mining analysis blocks (i.e., code snippets) from existing computational notebooks. In this case, the analysis blocks are used as the graph states, in place of their corresponding analysis steps. Figure 4 shows the general approach for mining analysis blocks into this recommendation graph. We extract the analysis blocks from existing computational notebooks and recover the transitions between states from the sequences observed in each notebook, with the weights signifying the frequency of observed transitions. Analysis blocks become nodes B_i in this graph, and edges represent probabilistic transitions $Pr(j|i) = P_{i,j}$, where the probabilities $P_{i,j}$ are taken from a stochastic matrix $\mathbb{P}$ that simply represents the frequency of transitions between blocks in the individual sequences.

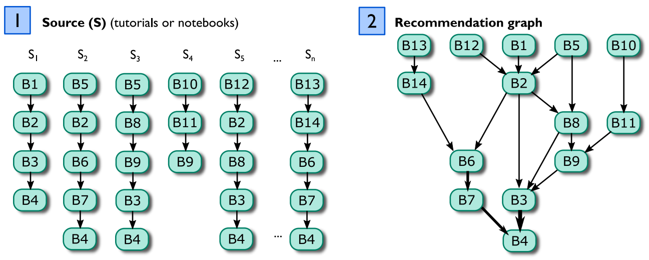


Figure 4. Mining blocks into a recommendation graph representing a Markov chain. In Step 1 (left), sources $S_1, \ldots, S_n$ (manually curated or automatically extracted) yield (ordered) sequences of blocks $S_i = (B_1, \ldots, B_m)$. In Step 2, a recommendation graph can be derived by matching blocks that appear in multiple sequences and joining the sequences at those nodes. Edges between blocks in the graph are the frequency-weighted state transitions in the chain.

To infer the full recommendation graph, we first construct a separate Markov chain for each notebook (or tutorial) identified as a source for our advisors. Specifically, we model each notebook as a Markov chain with one state per block and the transition probability to move from block B_i to the next block B_{i+1} for each time step (e.g., user input) expressed as $Pr(i+1|i) = 1$. Similar analysis steps are labeled with the same high-level identifier, representing a broader category of computation that transcends individual notebooks (e.g., $B1$, $B2$, etc. in Figure 4). The result is a larger two-dimensional nested list, where each notebook is one row within the list (i.e., the left side of Figure 4), and each column a sequence of analysis step categories.

We can then merge the resulting sequences into a single graph (e.g., merging S_1, S_2, etc. in Figure 4), and aggregate the relative frequencies associated with the different categories to determine transition weights (i.e., how often do we see blocks from category B1 executed before blocks from category B2?).

Specifically, the transition probability $P_{i,j}$ (and thus edge weight in the recommendation graph) for the i^{th} row and the j^{th} column is the number of edges from B_i to B_j across all the sequences, divided by the out-degree of B_i. In other words, the graph will have no edges (weight 0) between blocks that never appeared in sequence, and will have normalized weights for blocks that fan out to multiple different destinations (because they are used by many notebooks). To bootstrap the recommendation, we recommend the first analysis in all the sequences (the root nodes in the graph). To validate this method, we manually calculated the probabilities of two edges, each corresponding to the crowd and expert advisor. For example, we found that the recommendation of the expert advisors specified a probability of 33% for `group statistic` calculation transitioning to an `ANOVA-test`. Indeed, we found that this transition occurred 33% of the time in our sample of tutorials. The recommendation graph corresponding to our crowd advisors specified a probability of 7% for `matrix-normalization` calculations transitioning to the use of the `numpy-hstack` operation. We found this was accurate with a manual calculation of the probability of their co-occurrence.

Extracting Analysis Blocks for the Expert Advisor

We extracted analysis blocks for our expert advisor from online tutorials[†]. These tutorials were either Jupyter Notebooks or blogs which clearly delineated code from text. Analysis blocks correspond directly to code cells found in tutorial notebooks, or self-contained code snippets found in blog posts. We derived analysis blocks from these sources because we believed that it could sufficiently encapsulate current best practices in Python data science programming (D1).

While there exist many data science resources online, their focus and depth vary widely, from simple hands-on learning for beginners to expert-level guides on deep learning, sensitivity analysis, and model building and tuning. As a rule, we picked guides focused on teaching a specific analysis task (D2). We narrowed our search to end-to-end data science examples, which provide concrete sequences of analysis steps along the data science pipeline. Specifically, we selected examples that have an explicit purpose for the data analysis, step-by-step explanations and results, and runnable code. These requirements helped to ensure that the extracted analysis blocks will have similar types of functionality and were high-quality.

Formatting Analysis Blocks for the Expert Advisor

To ensure that the extracted analysis blocks are executable in Lodestar, we also apply a separate code curation process. From our experience, each source has a specific analysis goal, and the blocks across different sources may use different libraries, data attributes, and variables to achieve it. For example, a tutorial using the Boston housing dataset, may generate a scatter to examine a linear relationship between four housing attributes, while in a school test-scores dataset it only makes sense to examine a linear relationship in between two attributes. This is useful nuance for manual analysis, but cannot be directly used in a generic data analysis system

[†]Our supplemental materials includes a detailed report of the full process for extracting and curating analysis blocks for the expert advisor: https://osf.io/3gpsy/

such as Lodestar. In other words, the analysis blocks must be curated—typically generalized—to be applicable across multiple applications.

The block curation process is idiosyncratic, but consists of the following steps: (1) adding missing dependencies, (2) replacing data-specific labels and attributes, (3) setting appropriate default parameters, and (4) generalizing code to operate on general data frames and output data frames too. This process is very similar to our curation strategy for recommendations from our "crowd" advisor. We manually compared new blocks to exiting blocks within the library, to ensure there were no duplicates. Upon completion of the curation process, each new analysis block is added to the library for the recommendation graph.

Managing Analysis Blocks for the Crowd Advisor

We extracted analysis blocks for our crowd advisor from a corpus of approximately 6,000 Jupyter notebooks, originally collected by Rule et al.[42] We filtered out notebooks that did not contain import statements and API calls using common data science libraries, such as Numpy, Scikit-Learn, or Pandas. We first partition each notebook into discrete analysis blocks. For Jupyter notebooks, the code is often already partitioned by the notebook authors through the use of Jupyter notebook code *cells*. Our straightforward approach is to identify existing cells in the Jupyter notebook corpus as separate analysis blocks for Lodestar. Please see our supplemental materials for a detailed report on our full process for extracting and curating analysis blocks for the crowd advisor.

Our key insight for this process is that *similar data analysis steps often use similar API calls* in the code. For example, notebooks that leverage *sklearn.linear_model* to build a linear regression model using the *LinearRegression()* module could be characterized as performing the same analytical step. Using this idea, we construct a term vector to represent each analysis block, where the vector represents the normalized frequency of each API call that appears within the block. Each cell in the vector represents a unique API call observed in *any* notebook in the dataset, allowing the vectors for the block to be compared with any other block.

We use these term vectors to cluster the analysis blocks. Specifically, the normalized vectors are passed to a k-means clustering algorithm to be clustered for similarity. After some iteration, we identified 200 clusters as an ideal number for grouping the analysis blocks extracted from our corpus (please see our supplemental materials for more details). We observed cluster strength through the distribution of cells across clusters and through a silhouette score (score = 0.287). Given the underlying quality of the dataset, we found that 200 clusters were acceptable for manual processing. Each resulting cluster represents a set of analysis blocks that share similarities in functionality, and thus could also represent a shared or synonymous analysis step across the corresponding Jupyter notebooks.

Of the 200 representatives (one for each cluster), we ultimately selected 22 blocks as a starting set for the Lodestar library. For any given cluster, Lodestar needs a way of recommending a single analysis block to users. We use code-line count as a heuristic to pick a representative analysis blockfrom each cluster. Specifically, we pick the blocks which have a median number of lines relative to all other blocks within a cluster. We posit that this will yield the "average" code unit.

Blocks for both the crowd and expert advisors are formatted to follow the same consistent structure assumed by the Lodestar system. We format each analysis block to be a Python function, include necessary imports, convert the function's input and output to a data frame, and remove print statements and irrelevant comments.

Identifying Synonymous States Across Advisors

Of course, managing multiple advisors means that the system must track the state of the analysis in the recommendation graph for *all* advisors when the user selects a recommendation from a specific advisor. Our current solution uses a multi-level tagging mechanism where each block is manually tagged given its functionality; for example, a decision tree block could be tagged with `train-model` and `test-model`. Tags correspond to steps in the data analysis workflow. We developed an understanding of these steps using previous studies.[4,21,67] Much like Yan et al.,[67] we cast particular Python APIs to specific analysis steps. For example, Pandas *dropna* function was cast as a data-cleaning operation since dropping empty elements is a common way to clean data. Our tags include: `statistical-sampling`, `visualization`, `data-organization`, `data-cleaning`, `data-formatting` and `statistical-summary`.

In tagging analysis in this way, we allow for matching the new state of the specific advisor, chosen by the user, to relevant states in the other advisors. More specifically, if the user chooses a recommendation from the expert advisor that suggests running a specific decision tree block, the Lodestar engine will advance the crowd advisor to a state in its recommendation graph that corresponds to the `train-model` and `test-model` tags. This design, as well as ordering recommendations by probability ordering, allows Lodestar to guide best practices.

The same functionality is used when the user eschews all of the recommendations and instead selects directly from the library through the drop-down box in the recommendation panel. In this case, all of the advisor models will be advanced to the appropriate state matching the block that the user executed. This allows the user to iterate on techniques unhindered by a guided system.

Summative Evaluation

We conducted a second user study to evaluate the improvements we made to Lodestar after receiving formative feedback from the first user study. While the formative study evaluated the usability of the early prototype, this summative study evaluated the viability of Lodestar for learning data science practices.

The summative study was conducted over a period of one month. We interviewed 10 fledgling data scientists. 7 of these participants were new to Lodestar and 3 were part of our formative study. All participants were undergraduate university students who demonstrated knowledge of data science fundamentals through a university-level introductory course or other relevant experiences. Again, some students had experience with performing analysis on platforms like Excel and Tableau, but this was not a prerequisite for the recruitment process. Similar to the formative study, we chose undergraduate students for our user study population because they were learning data science principles for the first time. This study was approved by our home institution's IRB.

Method

Each interview lasted for 60 minutes and was divided into four phases. Unlike the formative study, this study was conducted exclusively online with video conferencing software. Prior to the interview, each participant gave us explicit consent to record audio and screen capture throughout the duration of the interview.

In our formative study, participants did not use Lodestar before completing the data exploration task, making it difficult to tease apart design challenges with the Lodestar prototype from a lack of user training. As a result, we included a separate training session where participants were given an overview of system features and then trained to use the system with an initial demo dataset. The training session was then followed by a think-aloud data exploration session that lasted for 15-20 minutes. Questions posed in the formative study can be found in the supplement. Afterward, participants were asked to verbally respond to a post-exploration questionnaire that assessed their view on the viability of the system. Questions included:

- What do you like about Lodestar? What do you dislike? Why?
- Would you use Lodestar outside of this study? Why/why not?
- If so, in what situations could you see yourself using Lodestar?

We analyzed participant responses from the think-aloud and questionnaire portion for themes regarding the usability of Lodestar. We present the participants' quotes which represent common themes.

Results

Here, we summarize both the strengths of our design as well as opportunities for future improvements as noted by participants. Identifiers for new participants begin with "N", whereas returning participants have the same identifiers as before.

Lodestar Strengths. **Intuitive and Supportive UI Features.** Many participants said that they found the (new) Lodestar interface design to be intuitive. For example, participant NP4 said they "*liked [a lot] of the different UI features like the tooltips and collapsing [views].*" Participant NP4 also liked that they were able to verify different analyses all on one page. FP2, FP3 and NP6 echoed this sentiment. Thus, the new learning widgets in Lodestar (e.g., tooltips and tabs) seem to help users learn how to use the interface and verify their work.

Integrates with Existing Tools and Workflows. Participants particularly liked the ability to export their workflow as a Jupyter Notebook file for editing outside of Lodestar. For example, NP4 said they "*like the integration with Jupyter Notebook and [the] exporting functionality.*"

Eases Data Science Tasks. NP3, NP4, FP2 and NP5 appreciated how Lodestar "*recommends what to do with the data, and based on that result, [recommends] something else*" (NP4). Overall, they found Lodestar helpful for guiding their analysis.

Lodestar also seemed to be helpful for specific data science tasks. For example, participants liked that Lodestar allowed them to quickly familiarize themselves with a particular dataset, which helped them determine what kinds of patterns or trends to analyze later on. This familiarization process is part of data profiling, an important and early task in the data science process.[22] Participants also valued the features in Lodestar for data exploration, for example participant FP3 said using Lodestar was "*Really convenient to do exploratory data analysis.*" Participant NP1 also stated that they would use Lodestar for "*specific cases where [they] don't know how to write the code.*" Participant NP6 and FP1 shared similar sentiments. These findings suggest that Lodestar helps users more easily complete data science tasks without being hindered by low-level programming issues, and may help users learn how relevant code could be written for future data science projects.

Limitations. **Customizing Visualization Outputs.** Many participants wanted the ability to customize and choose what attributes of the data were used in generating visualizations. For example, participant FP2 said that "*it would be nice to be able to set the parameters...*" Thus, even finer-grained control over visualization (and interaction) designs in Lodestar would be a point of improvement.

Additional Documentation. Some participants noted that providing more documentation of the features in Lodestar would be helpful when navigating the interface. For example, NP3 stated that it would be helpful if "*you could have [some information] about what the data is, where [it] came from, what the columns are.*" Thus, the Lodestar interface could be improved further to give users more context for the inputs to each analysis step.

Table 1. Participant demographics. These participants were all data science professionals.

Position Title	Age	Degree	Exp.
Machine Learning Engineer	30	C.S. B.S	2.6 yrs
Analytical Engineer	24	D.S. M.S	3.1 yrs
Data Scientist	34	D.S. M.S	2.4 yrs

Recommendation Evaluation

We conducted a third user study to evaluate the utility of Lodestar recommendations. We recruited three professional data scientists through a convenience sample of people within a professional network connected with the authors. Participant demographics can be found in Table 1. These professionals took part in an hour-long expert review[54] conducted using online videoconferencing. The interviewer initiated a video call with each participant and shared a screen with a running instance of Lodestar.

To start, participants were presented with two different analytical scenarios and asked to evaluate the strength of crowd and expert recommendations on behalf of an intern working within two scenarios. The first scenario proposed that an intern would be exploring the *cars* dataset and using Lodestar to understand the general trends. In the second scenario the intern would be trying to identify the factors which influence housing prices in the *boston-housing* dataset.

Participants were encouraged to remotely guide interactions with Lodestar recommendations and to build a data science workflow using Lodestar recommendations. At each step, after considering and choosing a recommendation, participants were asked to consider two questions:

- Why have they chosen this crowd or expert recommendation?
- Are the current crowd or expert recommendations appropriate for an intern performing the current task? Why or why not?

We adopted this protocol to ensure that participants have the flexibility to build an appropriate workflow for the objectives presented and the structure to provide feedback regarding a wide-array of analytical branches represented by the Markov recommender. Questions posed to the experts can be found in the supplement.

Results

We transcribed the verbal responses of each participant and coded their responses using the identifiers “crowd,” “expert,” and “workflow.” The *crowd* identifier classified reflections regarding crowd recommendations. The *expert* identified classified comments on expert recommendations. Finally, participants’ reflections on how the data science workflow was constructed and their interactions Lodestar were identified as comments regarding the *workflow*. We summarize the results of this analysis in this section. Identifiers for each professional participants begin with “P.”

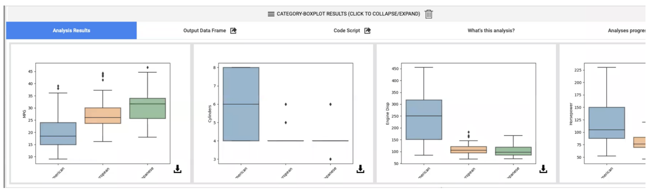
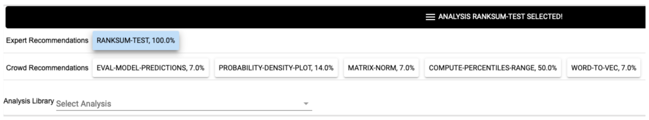


Figure 5. Visualization recommendations. Examples of visualizations generated by Lodestar recommendations implemented in Python and Seaborn.

Expert. All participants found that expert recommendations *“were quite reasonable”* (P1), and helpful for interns performing exploratory analysis as in the first scenario. Expert recommendations seemed to match expectations and act as a helpful guide for further analysis. P1 and P3 found the recommendations which visualized dataset attributes to be particularly helpful during exploration. For example, P1 said they found the *“Category Distribution” recommendation “almost too perfect”* (Figure 5).

While performing more directed analysis as part of the second scenario, participants found that expert recommendations were helpful during the beginning of the analysis. For example, P1 commented that *“looking at the data as the first step makes sense”*. However, P1 and P3 expressed a desire for more control over how the recommended techniques were being executed in order to dig deeper into data. All participants found the idea of exporting to a traditional notebook environment to be a useful next step in reaction to reaching the end of a Lodestar analytical track.



ANALYSIS RANKSUM-TEST SELECTED!

Expert Recommendations: RANKSUM-TEST, 100.0%

Crowd Recommendations: EVAL-MODEL-PREDICTIONS, 7.0% | PROBABILITY-DENSITY-PLOT, 14.0% | MATRIX-NORM, 7.0% | COMPUTE-PERCENTILES-RANGE, 50.0% | WORD-TO-VEC, 7.0%

Analysis Library: Select Analysis

Figure 6. Crowd advisor. The crowd advisor often recommended complex techniques.

Crowd. Participants found crowd recommendations overwhelming and unhelpful for exploratory tasks. The fact that the Lodestar displayed low confidence in the recommendations was a particularly strong deterrent for all three participants. For example, P2 said that *“there is one [crowd recommendation] I might have wanted to click but... the probability looks really low. So, I’m not really sure how to interpret that”* (Figure 6).

P1 and P2 found crowd recommendations equally unhelpful for the directed tasks of the second scenario. However, P3 felt that crowd recommendations could be occasionally helpful when expert techniques seem less varied. Unlike the other participants, P3 performed more than a handful of the advanced techniques suggested by the crowd: k-mean clustering, percentile range, and quantitative bar plots.

Workflow. Two participants found that expert and crowd recommendations were appropriately generated based on previous selections. However, these participants also found the data agnosticism of the recommendations to be confusing.

The system seem to the sufficiently transparent since all participants navigated to the “Code” tab in order to examine the programming details of each analytical step. However, P1 and P2 did not find the categorization of recommendations based on advisor (code source) to be helpful. Both P1 and P3 suggested categorizing recommendations in more ways to allow for better control over analytical goals. For example, P3 suggested separating recommendations geared towards visualizing a data attribute from recommendations which provide analytical support. This would be an interesting direction for our future work.

All three experts agreed that Lodestar recommendations would be helpful for novice users who were learning new analytical techniques and learning to program.

Summary. Participants found expert recommendations more appropriate for data exploration than crowd recommendations. This seems reasonable given that the tutorials we used to train our expert advisor were demonstrating exploratory data analysis. Participants generally found crowd recommendations difficult to trust and understand. However, P1 and P3 suggested that crowd recommendations occasionally supported directed analysis better than expert recommendations. Finally, participants expressed that a combination of expert and crowd recommendations would support interns who wanted to safely sandbox unfamiliar data analysis techniques.

Discussion

We have presented Lodestar, a computational notebook for rapid experimentation and learning of new data science practices. Instead of forcing fledgling analysts to search for and apply relevant data analysis methods by hand, Lodestar recommends suitable next steps for the current workflow using both manually curated as well as automatically crowd-sourced guidance. Our work on Lodestar has uncovered several interesting discussion points: the prospect for data science for novices, the actual “wisdom” of crowd recommendations, and alternate recommendation mechanisms.

Data Science for Non-Experts

The real power of Lodestar lies not in its data sources, which are publicly available to anyone online, but in its ability to synthesize the knowledge from these diverse sources into a single unified model. By sharing this knowledge in the form that data scientists are most familiar—Python (or R) source code—Lodestar provides reusable building blocks that can be transferred across workflows.

However, for the tool to be truly effective for its purpose, the library of analysis blocks must be expanded and drawn from a large set of sources. For example, new data sources could be incorporated to customize Lodestar for specific disciplines such as bio-informatics, computational journalism, and computer vision. Lodestar’s advisor model may be one way to support this; as suggested by our recommendation user study, instead of the “expert” vs. “crowd” dichotomy that our current implementation uses, a more robust implementation could support a plethora of pluggable advisors drawn from a central repository. In this way, the advisors, analysis blocks, and library could be community-driven and improved by anyone.

Choosing an analysis step or interpreting results in our current prototype still requires baseline data science knowledge, such as from a university data science course (indeed, all our participants had this). However, the Lodestar approach does alleviate lack of *expertise* in data science practice, which is often the case for academic learning.

The philosophy of the current Lodestar implementation is to give the user as many options as possible for how to proceed with the analysis. However, choice is sometimes a bad thing: for a novice data scientist, getting multiple—and, worse, conflicting—advice can be bewildering. In future work, it would be interesting to curate and coordinate recommendations from multiple advisors to help the user make better and more informed choices.

On the “Wisdom of the Crowd” for Data Analysis

While we are excited about the prospects of the “wisdom of the crowd”[52] for data science and analysis, it has become clear that this is an area that will require significantly more work. For example, our current approach is not entirely automated; manual curation is still required in choosing a representative block from the clustering analysis and in editing the block into the appropriate form that Lodestar expects, including eliminating side effects, removing output statements, and resolving dependencies. We plan to automate these steps in the future.

The need for manual curation, or at least review, is compounded by the fact that a significant portion of Rule et al.’s Jupyter corpus[42] was of low quality: some notebooks had cells with a single line of code, or all of the source code in a single cell. Many had non-functional code, syntax errors, or code that was never used. While we filtered such notebooks from our analysis, the signal-to-noise ratio in crowdsourced code is often low.

The remedy for many of these challenges can often be found in sheer scale. While we studied the “sampler” dataset containing 6,530 notebooks in this paper, the full 600 GB dataset contains more than 1.25 million notebooks. With access to this many examples, we could afford to discard more problematic ones. Furthermore, frequency of use would help ensure that best practices are easier to identify. Of course, a dataset of this size brings with it a new set of scalability challenges. Existing data processing[34] and code analysis[15,16] techniques could help address this big data challenge in the future.

Different Recommendation Strategies

The Lodestar recommendation engine is based on Markov chains, which are useful for representing a sequence of chained states or commands, as in a data science script. However Markov chains may oversimplify the relationships between analysis steps and data science users in some ways. It would be interesting to study how to use more sophisticated methods as part of the Lodestar recommendation engine. For example, state-of-the-art recommender systems tend to be organized into collaborative filtering, content-based filtering, and hybrid filtering.[1] Collaborative filtering is based on a social view of recommendation, where behavior by other users such as navigation, ratings, and their personal traits are used to match content to a specific user. In the case of Lodestar, this would enable the historical preferences of Lodestar users to guide other users. For content-based filtering, recommendations can be derived by comparing items to recommend with user preferences and auxiliary information. This approach could enable Lodestar users to be matched to specific analysis steps based on, e.g., workflows they have created in the past, specific data types, and metadata for existing datasets and code. Finally, we could combine methods to develop new hybrid recommendation strategies.

A recent development in artificial intelligence is to build recommender systems using deep learning techniques (or *deep recommenders*[3,68]), particularly for content-based approaches. Given our large available corpus of potential training data, unsupervised methods such as Recurrent Neural Networks could prove useful, since they are ideal for sequential data. The Lodestar advisor model provides a useful framework from which to incorporate and merge future recommendation strategies for data science. However, these topics are beyond the scope of this paper.

Limitations and Future Work

Our evaluation of the recommender suggests curated recommendations, geared towards different types of analytical goal (e.g. data cleaning, data exploration, visualization..etc.) can enable rapid experimentation with different programming techniques for the real-world. We limited our system to tutorials which presented high-level data explorations techniques. Advisors specifically geared to provide recommendations on data wrangling techniques could be the means to handle messy real-world datasets for Lodestar users. Advisors can even be designed to provide “unique” or less popular recommendations to ensure Lodestar users consider many options. Diversifying the recommendation techniques to target multiple goals will be an important part of future work.

When we first curated crowd techniques, our aim was to introduce users to a variety of more modern libraries and conventions—particularly since expert tutorials become outdated. The professional data science study participants, rightfully, were sometimes confused by the complexity of the techniques presented by the crowd advisor. We believe that more detailed documentation[60] would be helpful reducing this type of confusing.

Due to the many challenges of automatic code analysis, we currently do not allow users to write their own code directly in Lodestar, or even to modify existing code. To make online code editing possible, we would need an automatic classification process that could determine how new code fits into the recommendation graph so that the system could resume the analysis with new recommendations after manual code block. Such live updates to the recommender are not currently part of Lodestar, but are an interesting direction for future work. These live updates would provide a view into the "latest" analysis trends and enable a means for the Lodestar analysis library to grow in time. We anticipate that such a live update mechanism would feed into a dashboard by which Lodestar users could further view, label, and filter for valid analysis techniques.

We made several design decisions to the Lodestar notebook that will need to be revisited for a general implementation. Lodestar currently does not consider specifics about each input dataset while making recommendations—only display recommendations which do not programmatically fail to execute on the selected dataset. This should be studied in future work. Furthermore, all of our analysis blocks take a Pandas data frame as input, and generate a new data frame as output. Also, other disciplines use other data representations, and some computations may require passing multiple data objects as arguments. To address these limitations, we look to improving our existing design and thoroughly evaluating these improvements in our future work.

Finally, with the release of state-of-the-art Large Language Models (LLMs) from OpenAI, Microsoft, and Google, it is safe to say that the future of data science recommendation is changing rapidly. Plugins for these LLMs already exist that allow users to upload datasets and then ask for customized data science analysis using natural language queries. We think that the findings in this paper can help guide and influence these future directions.

References

1. Adomavicius G and Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17(6): 734–749. DOI: 10.1109/TKDE.2005.99. URL https://doi.org/10.1109/TKDE.2005.99.
2. Badam SK, Mathisen A, Rädle R, Klokmose CN and Elmqvist N (2019) Vistrates: A component model for ubiquitous analytics. *IEEE Transactions on Visualization and Computer Graphics* 25(1): 586–596. DOI:10.1109/TVCG.2018.2865144. URL https://doi.org/10.1109/TVCG.2018.2865144.
3. Bar El O, Milo T and Somech A (2020) Automatically generating data exploration sessions using deep reinforcement learning. In: *Proceedings of the ACM Conference on Management of Data*. New York, NY, USA: ACM, pp. 1527—-1537. DOI:10.1145/3318464.3389779. URL https://doi.org/10.1145/3318464.3389779.
4. Battle L and Heer J (2019) Characterizing exploratory visual analysis: A literature review and evaluation of analytic provenance in Tableau. *Computer Graphics Forum* 38(3): 145–159. DOI:10.1111/cgf.13678. URL https://doi.org/10.1111/cgf.13678.
5. Bertin J (1983) *Semiology of Graphics: Diagrams, Networks, Maps*. Madison, WI, USA: University of Wisconsin Press.
6. Bostock M and Heer J (2009) Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics* 15(6): 1121–1128. DOI:10.1109/TVCG.2009.174. URL https://doi.org/10.1109/TVCG.2009.174.
7. Bostock M, Ogievetsky V and Heer J (2011) D^3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17(12): 2301–2309. DOI:10.1109/TVCG.2011.185. URL https://doi.org/10.1109/TVCG.2011.185.
8. Cleveland WS and McGill R (1984) Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association* 79(387): 531–554. DOI:10.1080/01621459.1984.10478080. URL https://doi.org/10.1080/01621459.1984.10478080.
9. Conlen M and Heer J (2018) Idyll: A markup language for authoring and publishing interactive articles on the web. In: *Proceedings of the ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, pp. 977–989. DOI:10.1145/3242587.3242600. URL https://doi.org/10.1145/3242587.3242600.
10. Cui Z, Badam SK, Yalçin A and Elmqvist N (2019) DataSite: Proactive visual data exploration with computation of insight-based recommendations. *Information Visualization* 18(2): 251–267. DOI:10.1177/1473871618806555. URL https://doi.org/10.1177/1473871618806555.
11. Demiralp Ç, Haas PJ, Parthasarathy S and Pedapati T (2017) Foresight: Recommending visual insights. *Proceedings of the Very Large Database Endowment* 10(12): 1937–1940. DOI: 10.14778/3137765.3137813. URL https://doi.org/10.14778/3137765.3137813.
12. Drosos I, Barik T, Guo PJ, DeLine R and Gulwani S (2020) Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 1–12. DOI:10.1145/3313831.3376442. URL https://doi.org/10.1145/3313831.3376442.
13. Drozdal J, Weisz J, Wang D, Dass G, Yao B, Zhao C, Muller M, Ju L and Su H (2020) Trust in AutoML: Exploring information needs for establishing trust in automated machine learning systems. In: *Proceedings of the ACM Conference on Intelligent User Interfaces*. New York, NY, USA: ACM, p. 297–307. DOI:10.1145/3377325.3377501. URL https://doi.org/10.1145/3377325.3377501.
14. Epperson W, Jung-Lin Lee D, Wang L, Agarwal K, Parameswaran AG, Moritz D and Perer A (2022) Leveraging

analysis history for improved in situ visualization recommendation. *Computer Graphics Forum* 41(3): 145–155. DOI:10.1111/cgf.14529. URL https://doi.org/10.1111/cgf.14529.

15. Glassman EL, Scott J, Singh R, Guo PJ and Miller RC (2015) OverCode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction* 22(2): 7:1–7:35. DOI:10.1145/2699751. URL https://doi.org/10.1145/2699751.
16. Glassman EL, Zhang T, Hartmann B and Kim M (2018) Visualizing API usage examples at scale. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, p. 1–12. DOI:10.1145/3173574.3174154. URL https://doi.org/10.1145/3173574.3174154.
17. Head A, Jiang J, Smith J, Hearst MA and Hartmann B (2020) Composing flexibly-organized step-by-step tutorials from linked source code, snippets, and outputs. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, p. 1–12. DOI:10.1145/3313831.3376798. URL https://doi.org/10.1145/3313831.3376798.
18. Herlocker JL, Konstan JA, Terveen LG and Riedl JT (2004) Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22(1): 5–53. DOI:10.1145/963770.963772. URL https://doi.org/10.1145/963770.963772.
19. Hu K, Bakker MA, Li S, Kraska T and Hidalgo C (2019) VizML: A machine learning approach to visualization recommendation. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM. ISBN 9781450359702, p. 1–12. DOI:10.1145/3290605.3300358. URL https://doi.org/10.1145/3290605.3300358.
20. Javed W and Elmqvist N (2013) ExPlates: Spatializing interactive analysis to scaffold visual exploration. *Computer Graphics Forum* 32(3): 441–450. DOI:10.1111/cgf.12131. URL https://doi.org/10.1111/cgf.12131.
21. Kandel S, Paepcke A, Hellerstein J and Heer J (2011) Wrangler: Interactive visual specification of data transformation scripts. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 3363–3372. DOI:10.1145/1978942.1979444. URL https://doi.org/10.1145/1978942.1979444.
22. Kandel S, Paepcke A, Hellerstein JM and Heer J (2012) Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics* 18(12): 2917–2926. DOI:10.1109/TVCG.2012.219. URL https://doi.org/10.1109/TVCG.2012.219.
23. Kery MB, Ren D, Hohman F, Moritz D, Wongsuphasawat K and Patel K (2020) Mage: Fluid moves between code and graphical work in computational notebooks. In: *Proceedings of the ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, p. 140–151. DOI:10.1145/3379337.3415842. URL https://doi.org/10.1145/3379337.3415842.
24. Kim NW, Schweickart E, Liu Z, Dontcheva M, Li W, Popovic J and Pfister H (2017) Data-driven guides: Supporting expressive design for information graphics. *IEEE Transactions on Visualization and Computer Graphics* 23(1): 491–500. DOI:10.1109/TVCG.2016.2598620. URL https://doi.org/10.1109/TVCG.2016.2598620.
25. Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, Kelley K, Hamrick JB, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S and Willing C (2016) Jupyter notebooks – a publishing format for reproducible computational workflows. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Amsterdam, Netherlands: IOS Press, pp. 87–90. DOI:10.3233/978-1-61499-649-1-87. URL https://doi.org/10.3233/978-1-61499-649-1-87.
26. Knuth DE (1984) Literate programming. *The Computer Journal* 27(2): 97–111. DOI:10.1093/comjnl/27.2.97. URL https://doi.org/10.1093/comjnl/27.2.97.
27. Kross S and Guo PJ (2019) Practitioners teaching data science in industry and academia: Expectations, workflows, and challenges. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 1–14. DOI:10.1145/3290605.3300493. URL https://doi.org/10.1145/3290605.3300493.
28. Liu Z, Thompson J, Wilson A, Dontcheva M, Delorey J, Grigg S, Kerr B and Stasko J (2018) Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM. ISBN 9781450356206, pp. 1–13. DOI:10.1145/3173574.3173697. URL https://doi.org/10.1145/3173574.3173697.
29. Mackinlay J (1986) Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics* 5(2): 110–141. DOI:10.1145/22949.22950. URL https://doi.org/10.1145/22949.22950.
30. Mackinlay J, Hanrahan P and Stolte C (2007) Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics* 13(6): 1137–1144. DOI:10.1109/TVCG.2007.70594. URL https://doi.org/10.1109/TVCG.2007.70594.
31. Mathisen A, Horak T, Klokmose CN, Grønbæk K and Elmqvist N (2019) InsideInsights: Integrating data-driven reporting in collaborative visual analytics. *Computer Graphics Forum* 38(3): 649–661. DOI:10.1111/cgf.13717. URL https://doi.org/10.1111/cgf.13717.
32. Méndez GG, Nacenta MA and Vandenheste S (2016) iVoLVER: Interactive visual language for visualization extraction and reconstruction. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 4073–4085. DOI:10.1145/2858036.2858435. URL https://doi.org/10.1145/2858036.2858435.
33. Moritz D, Wang C, Nelson GL, Lin H, Smith AM, Howe B and Heer J (2019) Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. *IEEE Transactions on Visualization and Computer Graphics* 25(1): 438–448. DOI:10.1109/TVCG.2018.2865240. URL https://doi.org/10.1109/TVCG.2018.2865240.
34. Mudgal S, Li H, Rekatsinas T, Doan A, Park Y, Krishnan G, Deep R, Arcaute E and Raghavendra V (2018) Deep learning for entity matching: A design space exploration. In: *Proceedings of the ACM Conference on Management of Data*. New York, NY, USA: ACM, pp. 19–34. DOI:10.1145/3183713.3196926. URL https://doi.org/10.1145/3183713.3196926.
35. Perry DB, Howe B, Key AMF and Aragon C (2013) VizDeck: Streamlining exploratory visual analytics of scientific data. In: *Proceedings of the iConference*. Fort Worth, TX: iSchools, pp. 338–350. DOI:10.9776/13206. URL https://doi.org/10.

9776/13206.

36. Pirolli P and Card S (2005) The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In: *Proceedings of the International Conference on Intelligence Analysis*, volume 5. McLean, VA, USA: The MITRE Corporation, pp. 2–4.
37. Qian X, Rossi RA, Du F, Kim S, Koh E, Malik S, Lee TY and Chan J (2021) Learning to recommend visualizations from data. In: *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, p. 1359–1369. DOI:10.1145/3447548.3467224. URL https://doi.org/10.1145/3447548.3467224.
38. Rädle R, Nouwens M, Antonsen K, Eagan JR and Klokmose CN (2017) Codestrates: Literate computing with webstrates. In: *Proceedings of the ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, pp. 715–725. DOI:10.1145/3126594.3126642. URL https://doi.org/10.1145/3126594.3126642.
39. Ren D, Höllerer T and Yuan X (2014) iVisDesigner: Expressive interactive design of information visualizations. *IEEE Transactions on Visualization and Computer Graphics* 20(12): 2092–2101. DOI:10.1109/TVCG.2014.2346291. URL https://doi.org/10.1109/TVCG.2014.2346291.
40. Ren D, Lee B and Brehmer M (2019) Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics* 25(1): 789–799. DOI:10.1109/TVCG.2018.2865158. URL https://doi.org/10.1109/TVCG.2018.2865158.
41. Roth SF, Kolojejchick J, Mattis J and Goldstein J (1994) Interactive graphic design using automatic presentation knowledge. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, p. 112–117. DOI:10.1145/191666.191719. URL https://doi.org/10.1145/191666.191719.
42. Rule A, Tabard A and Hollan JD (2018) Exploration and explanation in computational notebooks. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 32:1–32:12. DOI:10.1145/3173574. URL https://doi.org/10.1145/3173574.
43. Russell DM, Stefik MJ, Pirolli P and Card SK (1993) The cost structure of sensemaking. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 269–276. DOI:10.1145/169059.169209. URL https://doi.org/10.1145/169059.169209.
44. Saket B, Kim H, Brown ET and Endert A (2017) Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE Transactions on Visualization and Computer Graphics* 23(1): 331–340. DOI:10.1109/TVCG.2016.2598839. URL https://doi.org/10.1109/TVCG.2016.2598839.
45. Satyanarayan A and Heer J (2014) Lyra: An interactive visualization design environment. *Computer Graphics Forum* 33(3): 351–360. DOI:10.1111/cgf.12391. URL https://doi.org/10.1111/cgf.12391.
46. Satyanarayan A, Moritz D, Wongsuphasawat K and Heer J (2017) Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics* 23(1): 341–350. DOI:10.1109/TVCG.2016.2599030. URL https://doi.org/10.1109/TVCG.2016.2599030.
47. Satyanarayan A, Russell R, Hoffswell J and Heer J (2016) Reactive Vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics* 22(1): 659–668. DOI:10.1109/TVCG.2015.2467091. URL https://doi.org/10.1109/TVCG.2015.2467091.
48. Seo J and Shneiderman B (2005) A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization* 4(2): 96–113. DOI:10.1057/palgrave.ivs.9500091. URL https://doi.org/10.1057/palgrave.ivs.9500091.
49. Siddiqui T, Kim A, Lee J, Karahalios K and Parameswaran A (2016) Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proceedings of the Very Large Database Endowment* 10(4): 457–468. DOI:10.14778/3025111.3025126. URL https://doi.org/10.14778/3025111.3025126.
50. Srinivasan A, Drucker SM, Endert A and Stasko J (2019) Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE Transactions on Visualization and Computer Graphics* 25(1): 672–681. DOI:10.1109/TVCG.2018.2865145. URL https://doi.org/10.1109/TVCG.2018.2865145.
51. Stolte C, Tang D and Hanrahan P (2002) Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* 8(1): 52–65. DOI:10.1109/2945.981851. URL https://doi.org/10.1109/2945.981851.
52. Surowiecki J (2004) *The Wisdom of Crowds: Why the Many are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*. New York, NY, USA: Anchor Books.
53. Tang B, Han S, Yiu ML, Ding R and Zhang D (2017) Extracting top-k insights from multi-dimensional data. In: *Proceedings of the ACM Conference on Management of Data*. New York, NY, USA: ACM, pp. 1509–1524. DOI:10.1145/3035918.3035922. URL https://doi.org/10.1145/3035918.3035922.
54. Tory M and Möller T (2005) Evaluating visualizations: Do expert reviews work? *IEEE Computer Graphics and Applications* 25(5): 8–11. DOI:10.1109/MCG.2005.102. URL https://doi.org/10.1109/MCG.2005.102.
55. Tufte E (2001) *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphic Press.
56. Ufford M, Pacer M, Seal M and Kelley K (2018) Beyond interactive: Notebook innovation at Netflix. https://medium.com/netflix-techblog/notebook-innovation-591ee3221233.
57. van den Elzen S and van Wijk JJ (2013) Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum* 32(3pt2): 191–200. DOI:10.1111/cgf.12106. URL https://doi.org/10.1111/cgf.12106.
58. VanderPlas J, Granger BE, Heer J, Moritz D, Wongsuphasawat K, Satyanarayan A, Lees E, Timofeev I, Welsh B and Sievert S (2018) Altair: Interactive statistical visualizations for Python. *The Journal of Open Source Software* 3(32). DOI:10.21105/joss.01057. URL https://doi.org/10.21105/joss.01057.
59. Vartak M, Madden S, Parameswaran A and Polyzotis N (2014) SeeDB: Automatically generating query visualizations. *Proceedings of the Very Large Database Endowment* 7(13): 1581–1584. DOI:10.14778/2733004.2733035. URL https://doi.org/10.14778/2733004.2733035.

60. Wang AY, Wang D, Drozdal J, Muller M, Park S, Weisz JD, Liu X, Wu L and Dugan C (2022) Documentation matters: Human-centered AI system to assist data science code documentation in computational notebooks. *ACM Transactions on Computer-Human Interaction* 29(2): 1–33. DOI:10.1145/3489465. URL https://doi.org/10.1145/3489465.
61. Wickham H (2016) *ggplot2: Elegant Graphics for Data Analysis*. New York, NY, USA: Springer. DOI:10.1007/978-3-319-24277-4. URL https://doi.org/10.1007/978-3-319-24277-4.
62. Wongsuphasawat K, Moritz D, Anand A, Mackinlay J, Howe B and Heer J (2016) Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics* 22(1): 649–658. DOI:10.1109/TVCG.2015.2467191. URL https://doi.org/10.1109/TVCG.2015.2467191.
63. Wongsuphasawat K, Qu Z, Moritz D, Chang R, Ouk F, Anand A, Mackinlay J, Howe B and Heer J (2017) Voyager 2: Augmenting visual analysis with partial view specifications. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 2648–2659. DOI:10.1145/3025453.3025768. URL https://doi.org/10.1145/3025453.3025768.
64. Wood J, Kachkaev and Dykes J (2019) Design exposition with literate visualization. *IEEE Transactions on Visualization and Computer Graphics* 25(1): 759–768. DOI:10.1109/TVCG.2018.2864836. URL https://doi.org/10.1109/TVCG.2018.2864836.
65. Xia H, Riche NH, Chevalier F, Araújo BRD and Wigdor D (2018) DataInk: Direct and creative data-oriented drawing. In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, pp. 223:1–223:13. DOI:10.1145/3173574. URL https://doi.org/10.1145/3173574.
66. Yalçin MA, Elmqvist N and Bederson BB (2017) Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics* 24(8): 2339–2352. DOI:10.1109/TVCG.2017.2723393. URL https://doi.org/10.1109/TVCG.2017.2723393.
67. Yan C and He Y (2020) Auto-Suggest: Learning-to-recommend data preparation steps using data science notebooks. In: *Proceedings of the ACM Conference on Management of Data*. New York, NY, USA: ACM, pp. 1539—-1554. DOI:10.1145/3318464.3389738. URL https://doi.org/10.1145/3318464.3389738.
68. Zhang S, Yao L and Sun A (2018) Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys* 52(1): 5:1–5:35. DOI:10.1145/3285029. URL https://doi.org/10.1145/3285029.