Buckaroo: A Direct Manipulation Visual Data Wrangler

Annabelle Warner
University of Utah
annabelle.warner@utah.edu

Andrew McNutt
University of Utah
andrew.mcnutt@utah.edu

Paul Rosen
University of Utah
paul.rosen@utah.edu

El Kindi Rezig
University of Utah
elkindi.rezig@utah.edu

ABSTRACT

Preparing datasets—a critical phase known as data wrangling—constitutes the dominant phase of data science development, consuming upwards of 80% of the total project time. This phase encompasses a myriad of tasks: parsing data, restructuring it for analysis, repairing inaccuracies, merging sources, eliminating duplicates, and ensuring overall data integrity. Traditional approaches, typically through manual coding in languages such as Python or using spreadsheets, are not only laborious but also error-prone. These issues range from missing entries and formatting inconsistencies to data type inaccuracies, all of which can affect the quality of downstream tasks if not properly corrected. To address these challenges, we present Buckaroo, a visualization system to highlight discrepancies in data and enable on-the-spot corrections through direct manipulations of visual objects. Buckaroo (1) automatically finds "interesting" data groups that exhibit anomalies compared to the rest of the groups and recommends them for inspection; (2) suggests wrangling actions that the user can choose to repair the anomalies; and (3) allows users to visually manipulate their data by displaying the effects of their wrangling actions and offering the ability to undo or redo these actions, which supports the iterative nature of data wrangling.

PVLDB Reference Format:
Annabelle Warner, Andrew McNutt, Paul Rosen, and El Kindi Rezig. Buckaroo: A Direct Manipulation Visual Data Wrangler. PVLDB, 18(12): 5423-5426, 2025.
doi:10.14778/3750601.3750687

1 INTRODUCTION

Data is ubiquitous, but it is also messy. Before data can be analyzed and acted upon, it must be prepared in a process commonly referred to as data wrangling. However, data wrangling is a well-documented bottleneck in the data science pipeline, accounting for as much as 80% of the time spent on data analysis [3, 4]. This process consists of multiple steps, including parsing the data (e.g., converting to a spreadsheet or loading into a database), transforming the data into the right set of rows and columns for analysis, cleaning invalid or missing data, integrating multiple data sources, removing duplicate or similar records, validating that the data is complete and correct, and so on [3, 9].

Data wrangling is time-consuming and error-prone—issues like missing values or formatting inconsistencies are hard to catch and can lead to incorrect results. Fixes are often applied inconsistently and non-reproducibly through spreadsheets or ad-hoc scripts.

Visualization for data wrangling. Visualization is widely recognized as a powerful tool for data analysis, enabling data scientists to identify patterns, trends, and relationships that might otherwise remain hidden in raw data. Interactive visual analytics systems allow users to dynamically manipulate data representations, facilitating exploration and accelerating the process of insight generation [5]. Successful applications of visualization have touched every area of science, engineering, business, and the humanities by making large and complex datasets more interpretable and actionable.

Visualization is frequently used in an ad-hoc manner during data wrangling to validate intermediate steps, identify anomalies, and check for correctness. Analysts generate summary statistics, histograms, scatterplots, or other graphics to inspect data distributions, detect missing values, find conflicting value definitions, and ensure transformations have been applied correctly [11]. This process is typically informal and reactive. While these checks can

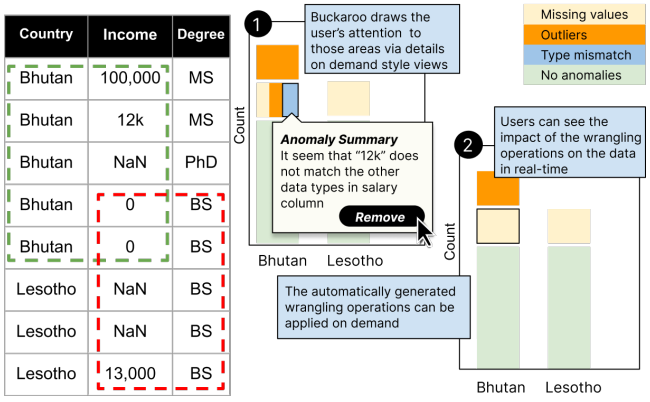

Figure 1: Motivating example: Both country groups (Bhutan and Lesotho) have anomalies such as outliers, missing values, and type mismatches. How to resolve these in an accessible way for non-technical users? Buckaroo visually highlights group anomalies and offers corresponding recommended repair operations. The user can then iteratively adjust their data visually until its reached a satisfactory clean state.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750687

be invaluable, their ad-hoc nature means they may be inconsistently applied, prone to oversight, and can make them difficult to reproduce. Moreover, because visualization tools are often separate from the wrangling process itself, switching between data transformations and visual validation can lead to a fragmented workflow where errors may slip undetected through the cracks between steps.

Example. Lou, a data scientist, is assigned to clean the dataset for further analysis as in Figure 1. Lou imports the dataset in Python and inspects it for inconsistencies. However, Lou faces several challenges due to the dataset's size: (1) Detecting subgroups with anomalies can be difficult since such errors are usually infrequent. For instance the income in subgroup `Country = Bhuthan` has a missing value and two entries with 0; (2) Correcting these anomalies is tricky, as fixes in one subgroup might introduce new issues in others (for example, removing all rows with an income of 0 from subgroup `Country = Bhuthan` would leave the subgroup `Degree = BS` with only a single income entry); and (3) The data cleaning process requires multiple iterations to develop an effective script.

If Lou instead utilizes Buckaroo (Figure 1❶), they would receive recommendations on which data groups with anomalies to inspect. Buckaroo enables Lou to choose specific data wrangling actions tailored to address different types of anomalies. Upon selecting an action, Lou can immediately see its impact (visually) on other groups, and check if it leads to new anomalies. This process is iterative, allowing Lou to apply or revert actions as needed. Once satisfied, Lou can use Buckaroo to generate a Python script of the wrangling steps for future use.

Centering the wrangling process in the visualization itself streamlines data preparation. It would reduce the reliance on iterative scripting and allow for more holistic and immediate error correction, enhancing both the efficiency and accuracy of Lou's work. With this approach, Lou could ensure the dataset is optimally prepared for analysis with less effort and fewer iterations.

Contributions. We introduce Buckaroo, an end-to-end system that enables data wrangling of tables by directly manipulating interactive charts. As illustrated in Figure 1, Buckaroo: (1) presents anomalous groups in an interactive chart (e.g., scatterplot, histogram, line chart) to visually expose those groups; (2) allows users to explore the various anomalies in the chart, and provides a summary of the anomalies in each group; (3) presents a list of actions (wrangling operations) to the user upon selecting a group on the chart to fix the anomaly; and (4) supports interactive visual data wrangling, so, users can go back and forth and observe the impact of one fix on other groups interactively until no anomalies are found.

Related work. Buckaroo bridges the gap between two lines of work: **(1) Anomaly detection and data cleaning:** Subgroup discovery is a well-established problem in data engineering [1, 6]. This line of work aims to identify interesting and interpretable subgroups within a dataset where the members of these subgroups exhibit given statistical properties that are significantly different from the general population. A related data wrangling demo is CoWrangler [3], where the goal is to recommend wrangling actions for a given dataset. Unlike those approaches, Buckaroo is orthogonal to the anomaly detection/repair that is being used. Therefore, Buckaroo can augment those methods by providing a direct manipulation

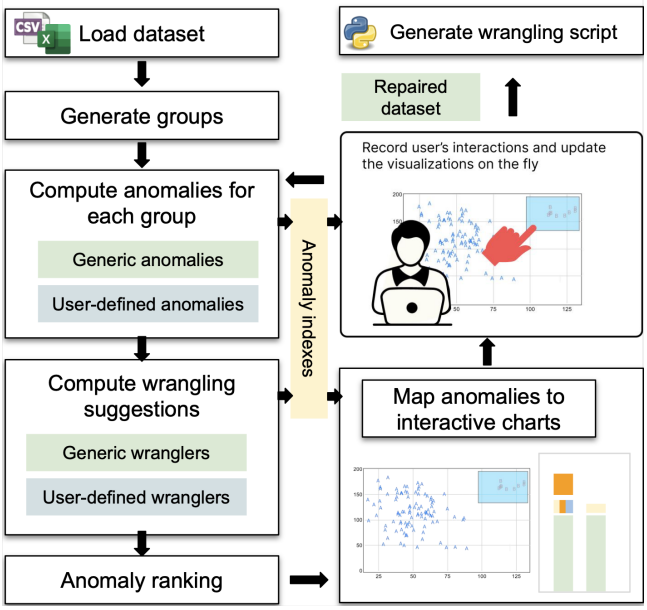

Figure 2: Workflow of Buckaroo: The user uploads a tabular dataset, and Buckaroo generates interactive charts for direct manipulation anomaly repair.

visual interface. **(2) Wrangling script visualization:** A variety of visualization systems have explored visual support for the data wrangling process. Xiong et al. [12] explore approaches for the visualization of wrangling scripts. We take the opposite tact, by interleaving wrangling activities in the visualizations themselves. Like us, Kandel et al. [7] synthesize data transformation scripts based on user intent expression in Wrangler (subsequently commercialized as Trifacta)—a strategy Chen et al. [2] more recently extend with natural language prompting. However, their system makes use of programming-by-demonstration as means of intent expression, whereas we center direct manipulation.

2 SYSTEM OVERVIEW

The main components of Buckaroo are illustrated in Figure 2. First, the user uploads a tabular dataset file (e.g., CSV, Excel spreadsheet). Then, Buckaroo generates all the groups formed by categorical attributes. For instance, from Figure 1, the group `Country = Bhuthan` is formed by the categorical attribute `Country`. Buckaroo then computes anomalies and wrangling suggestions for each group, ranks them by importance, and generates the interactive charts. Finally, Buckaroo records the actions taken by the user, and updates the charts when an action is performed.

2.1 Generating the groups

Buckaroo generates groups by doing a projection on numerical attributes and grouping by categorical attributes. For instance, in the motivating example (Figure 1), to get the subgroups containing the income for `country = Bhuthan`, we project `Income` and group by `Country`, and then extract the group corresponding to `Country = "Bhutan"`. Buckaroo allows users to specify the following:

- **Target attribute:** Users can specify which attributes in the table they are interested in, which would limit the number of generated groups. For instance, in Figure 1, the target attribute is `Income`.
- **Granularity specification:** Users can elect to specify the granularity of the generated groups using a minimum support parameter, which specifies the minimum number of data points per group.

2.2 Computing group anomalies

After computing the data groups, Buckaroo proceeds to detect the anomalies. Buckaroo (1) provides default anomaly types and their detector functions; and (2) exposes an API that allows users to override these functions or define custom anomaly types with their own detection logic.

Default anomaly types. By default, Buckaroo supports the following data anomalies: (1) **Missing values:** Missing values are common in tables. For each data group Buckaroo identifies the missing values; (2) **Outliers:** By default, outliers are those values that lie beyond 2 standard deviations from the all-group mean (i.e., a 95% confidence interval); (3) **Type mismatch:** This anomaly type is detected when non-numeric values appear in a numeric column; and (4) **Group data incompleteness:** Those are groups that have entries lower than a user-specified threshold (2 by default).

User-defined anomaly types. Users often have domain knowledge they want to apply during anomaly detection. For instance, a data scientist analyzing medical data may seek specific anomalies. Buckaroo supports this via an API that lets users define a custom `Detector` function, which flags anomalous tuples in a target column. Buckaroo treats these functions as black boxes, running them on each data group and collecting the flagged tuples.

Indexing anomalies. To support efficient retrieval, Buckaroo maintains two indexes: (1) from each anomaly type to the groups exhibiting it, and (2) from each group to its associated anomaly types.

2.3 Wrangling suggestions

After detecting anomalies for each data group, Buckaroo generates wrangling suggestions using default actions or user-defined wranglers.

Default wranglers. Buckaroo provides default actions for common anomalies, which users can override:

- **Missing values:** Suggests imputing with the group/column average or removing the row.
- **Outliers:** Recommends removal or imputation using group/column averages.
- **Type mismatch:** Detects if non-numeric values can be converted (e.g., "12k" to 12000), using LLM-generated functions.
- **Group incompleteness:** Suggests merging small groups with similar ones based on the distance between their vector embeddings (e.g., "USA" and "United States of America").

User-defined wranglers. For user-defined anomaly types, users provide an implementation of a `Wrangler` function for those anomalies. Each anomaly type has a unique identifier, which is then used to map a given wrangling function to an anomaly type.

2.4 Chart and Code Generation

Anomaly ranking. Using the indexes built in the anomaly detection step, Buckaroo selects the top-k (k is user-provided) groups that have the highest number of errors. By default, k is set to 3.

Generating the Charts. Buckaroo generates a chart matrix (Figure 3) where groups are shown in different boxes in the histogram bars. Users can visually inspect various types of group anomalies and directly manipulate them from the charts. Buckaroo uses the previously built indexes to show the anomaly types per group and the groups per anomaly type. Because it uses anomaly ranking, it can show only the top groups and anomaly types on the charts.

Recording the interactions. Buckaroo keeps track of all user actions, so they can undo and redo those actions interactively. This supports the notoriously iterative [10] process of data wrangling.

Code generation. Once the user is satisfied with the wrangling result, Buckaroo generates a stand-alone Python code that loads the input dataset, and includes the sequence of wrangling operations the user applied on the chart. Buckaroo uses the definitions of the `Detector` and `Wrangler` functions to produce this final script.

Implementation. Our demo system is built around a suite of D3-based charts (selected to allow for more bespoke interaction design). We use Arquero [8] for data processing because its straightforward DataFrame like structure scales well on client side tasks (supported by Apache Arrow), and because its small collection of analytic processing verbs have a straightforward mapping to SQL.

3 DEMONSTRATION PLAN

We demonstrate Buckaroo on the following real-world datasets: (1) Stackoverflow survey [1]; (2) Consumer complaints [2]; and (3) Chicago crime [3]. VLDB paricipants will wrangle those datasets to repair their group anomalies visually using Buckaroo. Figure 3 shows the main user interface of Buckaroo.

Demonstration outline. In this demo we aim to (1) show the participants how visual data wrangling can significantly make the data wrangling "fun" and less prone to errors; (2) allow the participants to experience the iterative nature of data wrangling through the suggested charts, where the participants can undo and redo actions as needed; and (3) thanks to its indexes, we show the participants how responsive Buckaroo is in updating the visualizations. In the following, we describe the demonstration scenario steps:

① **Uploading the dataset.** In Figure 3 Ⓐ, users start by uploading their dataset file. Buckaroo supports CSV and Excel formats.
② **Setting up anomaly detection.** After the dataset is uploaded, Buckaroo automatically detects and categorizes data errors, using colors to distinguish error types (Figure 3 Ⓑ). Buckaroo adopts a chart matrix to provide users with a comprehensive overview of data errors across multiple combinations of attributes. Users can visualize data errors in two modes:
Group name mode: Using a group-by attribute (Figure 3 Ⓒ), users can view the stacked histograms where the groups are color-coded

[1] https://survey.stackoverflow.co
[2] https://www.consumerfinance.gov/data-research/consumer-complaints
[3] https://www.chicago.gov/city/en/dataset/crime.html

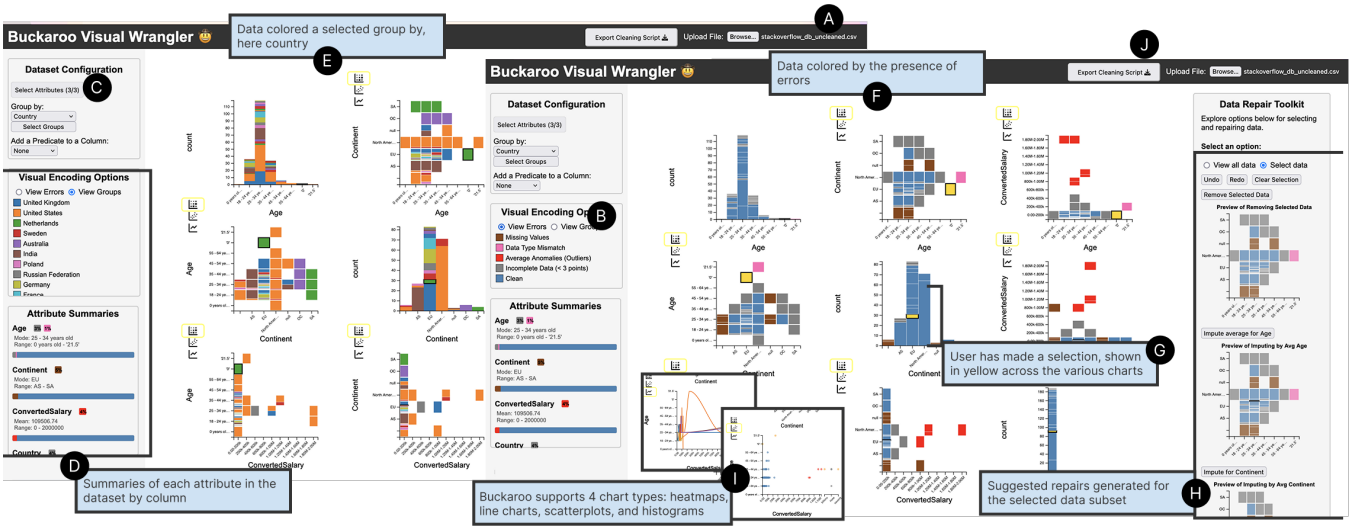

Figure 3: The UI of Buckaroo. Buckaroo features several grouping modes, i.e., by error type (F) or a group-by attribute (E). Buckaroo allows users to visualize errors and perform repairs by suggesting wrangling operations through a repair kit (H).

by values of the group-by column (Figure 3 E). Upon hovering on a given group, Buckaroo displays its error types.

Error type mode: In this mode (Figure 3 F), Buckaroo color-codes the groups (created from the group-by attribute) by their dominant error type.

③ **Attribute summary.** Buckaroo generates a list of recommended attributes to examine (Figure 3 D). For each attribute, it provides a summary of the associated error types and their frequency relative to the total number of values in that attribute. This feature is crucial to guide users to the most problematic groups.

④ **Repair kit.** When the user selects an error bar on the histogram (Figure 3 G), Buckaroo generates a list of recommended repairs for that error (Figure 3 H). The impact of each repair on the data is shown visually, allowing the user to assess which option is most appropriate. For example, the user might observe that a certain repair significantly alters the data distribution—typically an undesirable outcome. Buckaroo displays both built-in and user-defined wranglers to assist in repairing the input data.

⑤ **Visualization types.** Buckaroo supports heatmaps, line charts, scatterplots, and stacked histograms (Figure 3 I). These visualizations allow users to explore how different chart types affect the interpretability of data errors and their suggested repairs.

⑥ **Multi-step anomaly detection and wrangling.** Buckaroo is an interactive system that records all the user's actions, supporting undo/redo. We will guide the participants in a multi-step wrangling process where they can try out various repairs and be able to go back and forth until the dataset has been repaired.

⑦ **Code generation.** Finally, Buckaroo generates a Python script that captures all the wrangling actions applied to the dataset. This allows users to reapply the same transformations in the dataset for future use (Figure 3 J).

Demonstration engagement. In addition to a guided demonstration of Buckaroo, we will encourage participants to upload their own datasets to Buckaroo for a hands-on experience with visual data wrangling. This interactive session aims to highlight how in-visualization data wrangling represents a transformative shift in data preparation, which holds potential for further exploration.

REFERENCES

[1] Jakob Bach. 2025. Using Constraints to Discover Sparse and Alternative Subgroup Descriptions. arXiv:2406.01411 [cs.LG] https://arxiv.org/abs/2406.01411

[2] Wei-Hao Chen, Weixi Tong, Amanda Case, and Tianyi Zhang. 2025. Dango: A Mixed-Initiative Data Wrangling System using Large Language Model. (2025).

[3] Bhavya Chopra, Anna Fariha, Sumit Gulwani, Austin Z. Henley, Daniel Perelman, Mohammad Raza, Sherry Shi, Danny Simmons, and Ashish Tiwari. 2023. CoWrangler: Recommender System for Data-Wrangling Scripts *(SIGMOD '23)*.

[4] Dong Deng, Raul Castro Fernandez, Ziawasch Abedjan, Sibo Wang, Michael Stonebraker, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, and Nan Tang. 2017. The Data Civilizer System. In *CIDR*.

[5] Jeffrey Heer and Ben Shneiderman. 2012. Interactive Dynamics for Visual Analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue* 10, 2 (Feb. 2012), 30–55. https://doi.org/10.1145/2133416.2146416

[6] Franciso Herrera, Cristóbal José Carmona, Pedro González, and María José del Jesus. 2011. An overview on subgroup discovery: foundations and applications. *Knowl. Inf. Syst.* 29, 3 (2011), 495–525. https://doi.org/10.1007/s10115-010-0356-2

[7] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *SIGCHI Conference on Human Factors in Computing Systems*. 3363–3372.

[8] UW Interactive Data Lab. 2025. Arquero: JavaScript Library for Data Tables. https://idl.uw.edu/arquero/ Accessed: 2025-03-30.

[9] Zan Ahmad Naeem, Mohammad Shahmeer Ahmad, Mohamed Eltabakh, Mourad Ouzzani, and Nan Tang. 2024. RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes. *Proc. VLDB Endow.* 17, 12 (Aug. 2024), 4421–4424. https://doi.org/10.14778/3685800.3685890

[10] El Kindi Rezig, Lei Cao, Giovanni Simonini, Maxime Schoemans, Samuel Madden, Nan Tang, Mourad Ouzzani, and Michael Stonebraker. 2020. Dagger: A Data (not code) Debugger. In *CIDR*.

[11] Roy A Ruddle, James Cheshire, and Sara Johansson Fernstad. 2023. Tasks and visualizations used for data profiling: A survey and interview study. *IEEE Transactions on Visualization and Computer Graphics* (2023).

[12] Kai Xiong, Zhongsu Luo, Siwei Fu, Yongheng Wang, Mingliang Xu, and Yingcai Wu. 2022. Revealing the semantics of data wrangling scripts with COMANTICS. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2022), 117–127. https://doi.org/10.1109/TVCG.2022.3209470