Paper OS12

Block by Block: Introducing blockr, the Flexible Open-Source Dashboard Builder

Karma Tarap, Bristol Myers Squibb, Boudry, Switzerland
John Coene, Opifex, Geneva, Switzerland
Nicolas Bennet, David Granjon, Christoph Sax, Cynkra, Switzerland

ABSTRACT

This paper introduces *blockr*, an open-source R package designed to streamline the creation of data analysis pipelines and interactive dashboards. By providing a visual, block-based interface, *blockr* caters to both end-users and developers, bridging the gap between traditionally distinct roles in data science. We describe the architectural framework of *blockr*, explore its applications in clinical research, and examine its potential for broad adoption across various industries. The analysis demonstrates how *blockr* can streamline workflows, enhance collaboration, and democratize advanced analytics. A comparison with commercial software solutions, such as JMP Clinical and Spotfire, underscores the accessibility and flexibility of this platform in rapidly evolving research environments.

BACKGROUND AND MOTIVATION

The field of data analysis, particularly in clinical research, is experiencing rapid evolution driven by increasing data complexity and the demand for more agile, responsive analytical tools. Traditional approaches to dashboard development often create bottlenecks by separating end-users, such as statisticians and clinicians, from technical developers. This separation, coupled with rigid structures and long development cycles, delays the analytical process and limits researchers' ability to dynamically explore data and generate timely insights. As a result, researchers frequently find themselves constrained by predefined analyses, unable to quickly adapt their approach as new questions arise in fast-paced research environments.

While *R* [1] is a powerful tool for statistical analysis and data manipulation, creating interactive, web-based dashboards in R typically requires advanced knowledge of frameworks such as *Shiny* [2]. This technical barrier poses a significant challenge for many statisticians, programmers, and data scientists.

Moreover, ensuring reproducibility in data analysis has become increasingly critical, particularly in regulated fields such as clinical research. Many current tools lack built-in mechanisms for code export, making it challenging to replicate analyses.

In response to these challenges, we introduce *blockr* [3], a novel open-source platform that employs a block-based visual interface for the creation and manipulation of analytical pipelines. blockr bridges the gap between R's powerful statistical capabilities and the need for a more intuitive, flexible interface for data exploration and visualization. The platform caters to both end-users and developers: it provides a user-friendly interface for constructing analysis pipelines without requiring coding knowledge, while offering developers a robust framework for extending functionality through custom blocks. This dual focus democratizes the analytical process, allowing researchers to take control of their data analysis and enabling developers to enhance the system's capabilities efficiently. Moreover, by providing built-in mechanisms for code export, blockr ensures that analyses can be reproduced and versioned, thus supporting the stringent requirements for validated environments and addressing the critical challenge of reproducibility in data analysis.

ARCHITECTURE AND IMPLEMENTATION

The architecture of blockr is designed to balance user flexibility with developer accessibility, built around three primary components: blocks, stacks, and workspaces. Each block represents a specific operation, such as data transformations or visualizations, functioning as nodes within a Directed Acyclic Graph (DAG) that models the flow of data through the analysis pipeline. Blocks are modular and reusable, supporting the construction of complex analytical workflows without requiring users to code.

A critical feature is the Block Registry, which acts as a "supermarket" of reusable blocks. Developers can create custom blocks and then register them within the system, making these blocks accessible across different projects. This registry facilitates collaboration and efficiency, enabling users and developers to leverage previously developed blocks and reduce redundant work.

Additionally, blockr supports contextual blocks, which ensures that users are presented with only the blocks compatible with their current workflow. For example, when a user adds a new block to their stack, only relevant blocks for that stage of the pipeline are suggested—such as data transformation blocks for data operations or visualization blocks when plotting. This feature enhances the user experience by simplifying block selection and ensuring that pipelines are constructed without errors or irrelevant steps.

Figure 1: Overview of UI and individual blocks in {blockr}.

Stacks are sequences of interconnected blocks, forming complete analytical workflows. Users can modify these stacks by adding, removing, providing them with the flexibility to adapt their analysis to specific needs. This ability to customize the analytical process at a granular level enhances the system's overall versatility.

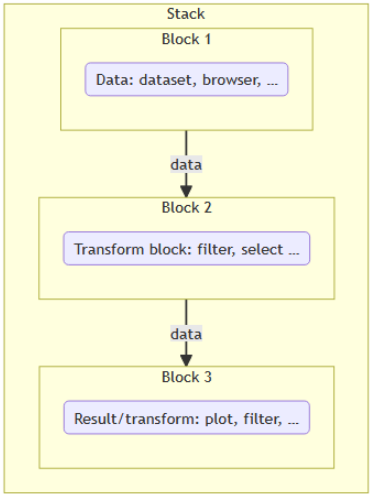

Figure 2: Representation of a stack as a linear combination of blocks.

Workspaces serve as containers for multiple stacks, allowing for the organization of complex, multi-stage analyses. This design enables users to manage and switch between different analytical tasks within a single project. In addition,

workspaces support interactions between stacks, facilitating more complex operations, such as joining datasets from different sources.

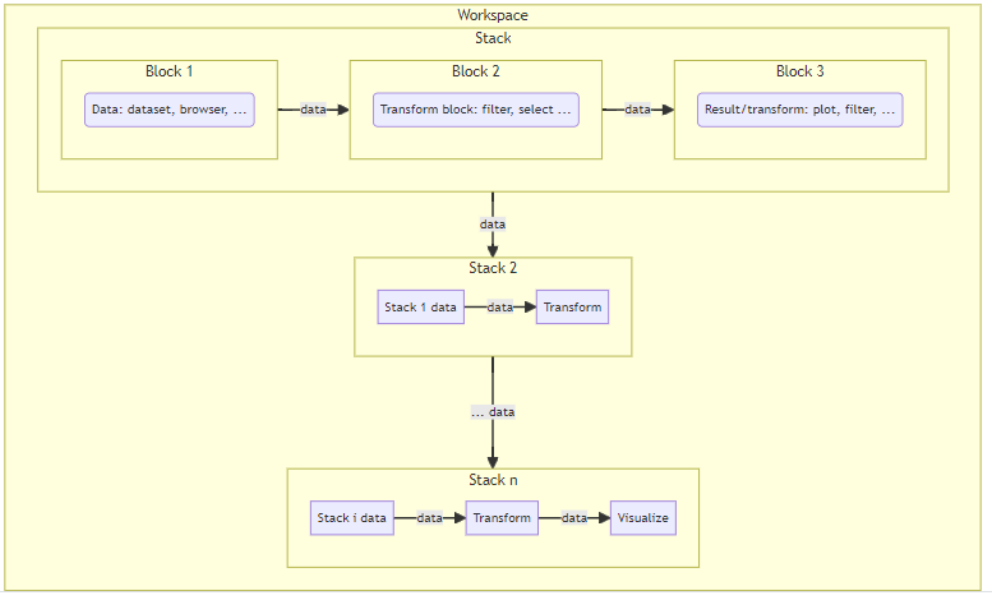

Figure 3:Representation of a workspace as a collection of stacks.

Technically, blockr leverages non-standard evaluation techniques within the R ecosystem to simplify block creation and manipulation. A custom Shiny server generation system translates block-based configurations into reactive Shiny applications, abstracting much of the complexity inherent in Shiny app development. Furthermore, blockr dynamically generates R code based on the blocks used in the analysis, ensuring that analyses can be exported as reproducible, standalone R scripts. This feature is particularly valuable in clinical research, where reproducibility is often a regulatory requirement.

The platform is also embeddable into existing R Shiny applications, allowing organizations to enhance their analytical capabilities without overhauling their current systems. This integration capability ensures that it can fit into established data infrastructure setups.

EXTENSIBILITY FOR DEVELOPERS

A cornerstone of blockr's architecture is its extensibility, which allows developers to expand the platform's capabilities with minimal effort. Using a declarative syntax, developers can define new analytical blocks by specifying their inputs, outputs, and computational logic. This approach focuses on statistical and data transformation functionalities, allowing developers to extend the platform without dealing with web interface design complexities.

To illustrate this, consider the creation of a new linear model block (new_lm_block). In this example, the block allows users to define a linear regression model through the interface. The UI fields are specified, and the computational logic is captured in the expr field, which stores the code to be evaluated when the block is executed or exported for reproducibility.

```
new_lm_block <- function(y = character(), predictor = character(), ...) {
        all_cols <- function(data) colnames(data)

        fields <- list(
                y = new_select_field(y, all_cols, type = "name"),
                predictor = new_select_field(predictor, all_cols, type = "name")
        )
        new_block(
                fields = fields,
                expr = quote({
                        model <- lm(data = data, formula = .(y) ~ .(predictor))
                        broom::tidy(model)
                }),
        ...,
        class = c("lm_block", "transform_block")
)
```

In this block the UI fields are defined in the fields list. This allows the user to select variables from the dataset for the dependent variable (y) and the predictor (predictor) through dropdown selectors, which are created using new_select_field.

The computational logic is specified in the expr field, which stores the R code that will be evaluated when the block is executed. In this case, the code fits a linear model (lm) using the selected variables and produces a tidy summary with the broom::tidy function. The expr field is also critical for reproducibility, as it allows the block's code to be exported, ensuring that the analysis can be replicated exactly.

Once the block is defined, it must be registered to make it reusable across different analyses:

```
register_lm_block <- function(pkg) {
        register_block(
                constructor = new_lm_block,
                name = "lm block",
                description = "Create a linear model block",
                classes = c("lm_block", "transform_block"),
                input = "data.frame",
                output = "data.frame",
                package = pkg
        )
}
```

This registration adds the newly created block to the Block Registry, making it available for reuse in future projects. This approach promotes modularity and reuse by allowing other users and developers to leverage pre-existing blocks and extend blockr's capabilities.

The separation of UI elements and computational logic in blockr's declarative syntax enables developers to focus on defining analytical functionality, while the platform handles the interface and reproducibility requirements. This extensibility fosters a growing ecosystem of reusable blocks, which can be shared across research teams, ensuring that blockr remains adaptable to emerging analytical needs.

APPLICATIONS IN CLINICAL RESEARCH AND BEYOND

The block-based interface of blockr has demonstrated exceptional value in the context of clinical research, where the complexity of datasets and the stringent need for reproducibility present significant challenges. One of the key strengths of blockr is its ability utilize existing R packages like *Cardinal* [4] to rapidly create clinical tables and outputs (Figure 4).

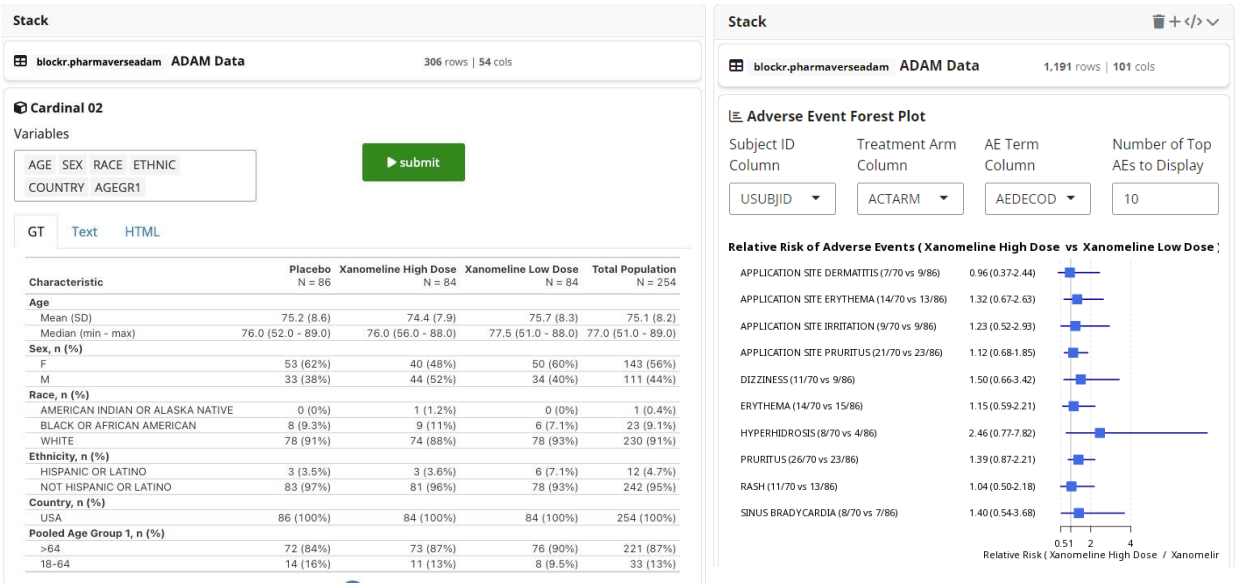

Figure 4: Demographic Summary via the Cardinal Package. Custom Forest Plot.

Beyond visualizations, blockr excels in creating data transformations that are foundational for clinical research workflows. Users can build data pipelines with common transformations such as filtering, mutating, and merging datasets or defining custom operations like calculating changes from baseline.

Another powerful feature of blockr is the integration of *ggplot* [5] layers as blocks, which gives users the flexibility to create sophisticated data visualizations while leveraging the full expressive power of the *grammar of graphics* [6]. By stacking ggplot layers as blocks, users can iteratively build plots—such as survival curves, bar charts, or scatter plots—customizing them at each step without writing raw ggplot2 code.

While blockr has its roots in clinical research, its flexible architecture and modular blocks make it highly adaptable to other industries. For example, in finance, blockr can be employed to construct tools for portfolio analysis, risk assessment, and market trend visualizations. An example of this is a stock price forecasting pipeline that integrates external libraries such as *quantmod* [7] and Prophet [8] to fetch stock data, perform time series analysis, and generate price forecasts (Figure 7). Additionally, blockr can be used for Marketing Impact Analysis, providing dynamic dashboards to assess campaign performance using *CausalImpact* [9] (Figure 8).

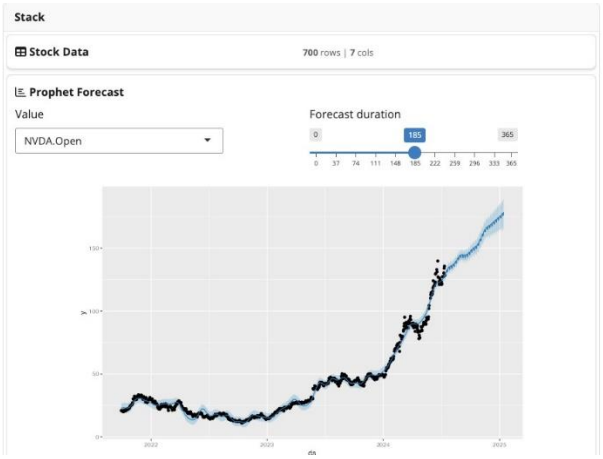

Figure 7: Stock price example using fbprophet

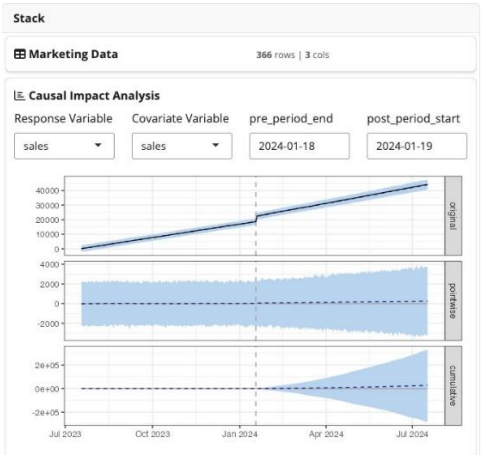

Figure 8: Marketing campaign analysis using CausalImpact

COMPARATIVE ANALYSIS WITH COMMERCIAL SOFTWARE

Compared to commercial software like JMP Clinical [10] and Spotfire [11], blockr offers a unique combination of flexibility and accessibility. JMP Clinical and Spotfire provide robust features for clinical data analysis and powerful visualization capabilities, respectively, but often come with high licensing costs and steep learning curves. In contrast, blockr is freely available as an open-source tool and highly customizeable.

While commercial solutions offer comprehensive support, regular updates, and polished interfaces, blockr may require internal resources for maintenance, custom development, and support. Organizations must weigh blockr's cost savings and flexibility against the convenience and robustness of commercial alternatives.

blockr offers a compelling option that can coexist with or complement commercial tools, providing organizations with a broader spectrum of analytical solutions. The choice between blockr and commercial software will depend on an organization's specific requirements, technical capabilities, and long-term data analysis strategy.

CONCLUSION

blockr offers a significant improvement in open-source data analysis by simplifying the creation and modification of analytical pipelines. Its block-based interface allows non-programmers to explore data independently, while developers benefit from an efficient framework for extending functionality. The open-source nature of blockr ensures continuous development, with new features driven by community contributions.

Its integration capabilities allow seamless adoption into existing infrastructures, making it a flexible tool for diverse industries. The platform's full potential will depend on the continued growth of its user and developer community, which will be essential for maintaining innovation and expanding the ecosystem of reusable blocks.

REFERENCES

1. **R**
   - R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/

2. **Shiny**
   - Chang, W., Cheng, J., Allaire, J. J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2023). *Shiny: Web Application Framework for R*. URL: https://CRAN.R-project.org/package=shiny

3. **Blockr**
   - Bristol-Myers Squibb. (2024). *blockr: Composable, extensible no-code UI*. GitHub Repository. URL: https://github.com/BristolMyersSquibb/blockr

4. **Cardinal**
   - Pharmaverse. (2024). Cardinal: FDA Safety Tables and Figures. Pharmaverse Initiative. Available at URL:https://pharmaverse.github.io/cardinal/ and URL:https://github.com/pharmaverse/cardinal.

5. **ggplot**
   - Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. URL: https://ggplot2.tidyverse.org

6. **Grammar of Graphics**
   - Wilkinson, L. (2005). *The Grammar of Graphics*. Springer Science & Business Media.

7. **quantmod**
   - Ryan, J. A. (2020). *quantmod: Quantitative Financial Modelling Framework*. URL: https://CRAN.R-project.org/package=quantmod

8. **Prophet**
   - Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37-45. URL: https://facebook.github.io/prophet/

9. **Causal Impact**
   - Brodersen, K. H., Gallusser, F., Koehler, J., Remy, N., & Scott, S. L. (2015). Inferring causal impact using Bayesian structural time-series models. *The Annals of Applied Statistics*, 9(1), 247-274. URL: https://google.github.io/CausalImpact/

ACKNOWLEDGMENTS

The development of blockr was made possible through a collaborative partnership between cynkra and Opifex, with sponsorship from Bristol-Myers Squibb (BMS).

We would like to acknowledge the contributions from the following individuals in no particular order:

Samar Noor (BMS), Nicole Thorne (BMS), Rashad Rasul (BMS), Oriana Esposito (BMS), Silvia Mosulen Machuca (BMS), Ramnath Dhadage (Ephicacy).

CONTACT INFORMATION

Karma Tarap
Bristol Myers Squibb, Boudry, Switzerland
Email: karma.tarap@bms.com
Website: www.bms.com

John Coene
Opifex, Geneva, Switzerland
Email: john@opifex.org
Website: https://opifex.org/

Nicolas Bennet
Cynkra, Zurich, Switzerland
Email: nicolas@cynkra.com
Website: https://cynkra.com

David Granjon
Cynkra, Zurich, Switzerland
Email: david.granjon@cynkra.com
Website: https://cynkra.com

Christoph Sax
Cynkra, Zurich, Switzerland
Email: christoph@cynkra.com
Website: https://cynkra.com