Vizagrams.jl: A Julia-based visualization framework that integrates diagramming and data visualization

Davi Sales Barreira [1¶], **Asla Medeiros e Sá** [2], **and Flávio Codeço Coelho** [1]

1 Fundação Getúlio Vargas - School of Applied Mathematics, Brazil **2** IMPA Tech, Brazil ¶ Corresponding author

DOI: 10.21105/joss.09037

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @jolars
- @abhishektiwari

Submitted: 23 July 2025
Published: 15 October 2025

Summary

Vizagrams.jl is a Julia (Bezanson et al., 2017) package that integrates diagramming and data visualization into a single framework. It implements a diagramming domain-specific language (DSL) together with a visualization grammar, thus allowing users to create diagrams by combining and transforming plots, as well as to create new visualization specifications using diagramming operations. The package aims to be able to produce complex, highly customizable visualizations while also providing intuitive usability.

Statement of need

A fundamental challenge for every data visualization tool is balancing abstraction and expressiveness. Expressiveness consists in the ability of a framework to produce a variety of visualizations, while abstraction consists in simplifying descriptions by omitting details, thus improving usability. Some tools prioritize high expressiveness at the expense of abstraction (e.g., D3.js (Bostock et al., 2011)), while others prioritize abstraction over expressiveness by focusing on ready-to-use chart recipes (e.g., Matplotlib (Hunter, 2007), Seaborn (Waskom, 2021), Makie.jl (Danisch & Krumbiegel, 2021), Plots.jl (Breloff, 2025)).

A possible solution to this challenge is to use visualization grammars. These grammars define a structured set of rules for specifying visualizations, allowing users to construct a wide range of plots by adjusting the components of a specification, rather than relying on a fixed set of chart types or intricate manual constructions (Mei et al., 2018).

Wilkinson's Grammar of Graphics (GoG) (Wilkinson, 2012) is one of the most influential theoretical contributions to visualization grammars, and has significantly influenced the design of various grammars (Li et al., 2020; LY'i et al., 2021; Pu & Kay, 2020; Satyanarayan et al., 2014, 2016; Wickham, 2011a; Wickham & Wickham, 2007). Despite their success, visualization grammars still have limitations in terms of expressiveness, particularly when dealing with complex visualizations such as those containing custom marks, composite graphics or intricate layouts.

Vizagrams.jl addresses these limitations by implementing a novel theoretical framework that integrates diagramming and data visualization. It does this by creating a visualization grammar over a diagramming DSL, where plots are treated as graphical objects, and diagramming operations can be used within graphic specifications, thus interweaving both concepts. The package is particularly valuable for data scientists, academic researchers, statisticians and any other user who needs to create complex visualizations and possesses programming knowledge. Its diagramming capabilities also make it a valuable tool for educational purposes, such as illustrating mathematical concepts. Vizagrams.jl has been developed as part of the PhD thesis of the first author (Barreira, 2025), where the theory behind its implementation has been explained in detail.

Related works

Vizagrams.jl's diagramming DSL was inspired by the work of Yorgey (Yorgey, 2012), who provided the theoretical foundation for diagram construction used by the Haskell Diagrams library (Yates & Yorgey, 2015). The Haskell Diagrams library has also inspired other diagramming libraries such as Diagrammar (Granström, 2022) and Compose.jl (Giovine, 2022). There are also diagramming libraries not based on Yorgey's conception, such as Bluefish (Pollock et al., 2024) and Penrose (Ye et al., 2020).

Besides Vizagrams.jl, other packages have used the idea of constructing data visualization tools on top of diagramming frameworks. For example, the Gadfly.jl (Jones et al., 2021) visualization package is built on the diagramming package Compose.jl (Giovine, 2022). Yet, Gadfly.jl implements a grammar similar to ggplot2 (Wickham & Wickham, 2007), rather than attempting to integrate diagramming and data visualization.

When it comes to visualization grammars, Vizagrams.jl implements a graphic specification based on the one used by Vega-Lite (Satyanarayan et al., 2016), which is a high-level visualization grammar built on top of Vega (Satyanarayan et al., 2014). The landscape of visualization grammars encompasses low-level implementations that prioritize expressiveness (e.g., Vega (Satyanarayan et al., 2014)) and high-level implementations that prioritize user-friendly specifications (e.g., Vega-Lite (Satyanarayan et al., 2016), ggplot2 (Wickham, 2011a), Polaris (Stolte et al., 2002), Observable Plot (*Observable Plot*, 2024), Atlas (Liu et al., 2021), AlgebraOfGraphics.jl, Gramm (Morel, 2018)). There are also specialized visualization grammars, such as Gosling (LY'i et al., 2021) for genomics data visualization, GoTree (Li et al., 2020) for visualizations with tree layouts, PGoG (Pu & Kay, 2020) for visualizations depicting probabilities, ggalluvial (Brunson, 2020) for plotting alluvial diagrams.

Besides diagramming and visualization grammars, there are also visualization authoring systems. These are software platforms or tools that allow users to create visualizations without the need for programming knowledge. Examples of visualization authoring systems are Improvise (Weaver, 2004), Data Illustrator (Liu et al., 2018), Charticulator (Ren et al., 2018), and StructGraphics (Tsandilas, 2020). These systems enable users to design and manipulate custom graphical marks, though they do so within graphical user interfaces rather than embedded domain-specific languages.

Overview of functionality

The main features of Vizagrams.jl are:

- Diagramming DSL
- Custom graphical marks
- Visualization grammar for creating data visualizations
- Graphic expressions to create new data visualization specifications
- Ability to manipulate and combine plots using diagramming operations

Diagrams are built by composing graphical marks and applying graphical transformations. Graphical marks are objects that can be drawn on the screen, while graphical transformations are actions such as translation, rotation, change of color, and so on. The package provides a set of ready-to-use marks, such as `Circle`, `Square`, `Line`, `TextMark`, but users can extend this by creating their own marks.

The main operations for building diagrams are:

- `+` for composing marks and diagrams, e.g., `d1 + d2` creates a new diagram where `d1` is rendered first, followed by `d2`
- `T(x,y)` for translating
- `R(θ)` for rotating
- `U(x)` for uniform scaling
- `M(p)` for reflecting
- `S(:attribute=>value)` for setting style attributes
- `*` for applying transformations, e.g., `T(x,y) * d` translates the diagram `d` by `(x,y)`

The following example shows how to create a simple diagram:

```
using Vizagrams
# Composing marks
d = Circle() + T(2,0)*Square()
# Composing previous diagram with a new mark
d = S(:fill=>:white,:stroke=>:black)*Circle(r=2) + Line([[0,0],[3,0]]) + d
draw(d)
```

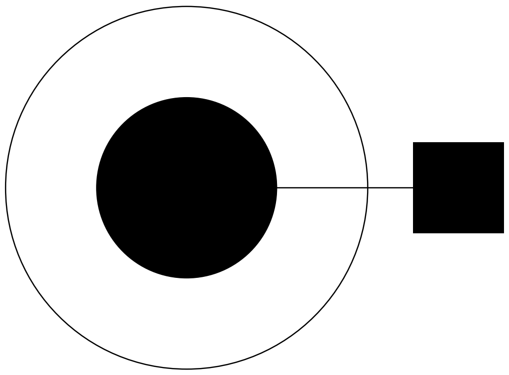

Figure 1: A simple diagram.

For creating data visualizations, Vizagrams.jl provides a visualization grammar with a syntax similar to Vega-Lite (Satyanarayan et al., 2016). Given a dataset, the user can create a visualization by mapping columns to visual properties, and specifying the `graphic` to be used (akin to what Vega-Lite calls `mark`). This specification is implemented internally as a mark `Plot`, thus making plots manipulable as any other mark.

Below is an example of a data visualization created with the visualization grammar. We use the `cars` dataset from the VegaDatasets.jl package together with the DataFrames.jl package (Bouchet-Valat & Kamiński, 2023).

```
using DataFrames
using VegaDatasets
df = DataFrame(dataset("cars"));
df = dropmissing(df);
simple_plot = Plot(
    data=df,
    encodings=(
        x=(field=:Horsepower,),
        y=(field=:Miles_per_Gallon,),
        color=(field=:Origin,),
    ),
    graphic=Circle(r=5)
)
draw(simple_plot)
```

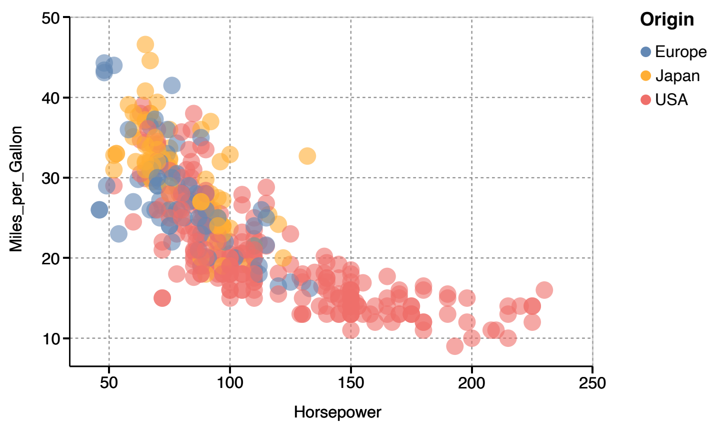

Figure 2: Scatter plot created with the visualization grammar.

One of the ways Vizagrams.jl extends visualization grammars is by allowing the use of graphic expressions inside the `graphic` component. A graphic expression is a function that takes a dataset and constructs a diagram using a pattern similar to the split-apply-combine strategy from Wickham (Wickham, 2011b). The following example shows how to use a graphic expression with our previously created diagram `d`:

```
complex_plot = Plot(
    data=df,
    encodings=(
        x=(field=:Horsepower,),
        y=(field=:Miles_per_Gallon,),
        color=(field=:Origin,),
        size=(field=:Weight_in_lbs,),
        rotate=(field=:Acceleration,),
    ),
    graphic=∑() do row
        T(row[:x],row[:y])*
        U(row[:size])*
        S(:fill=>row[:color],:opacity=>0.5)*
        R(row[:rotate])*
        d
    end
)
draw(complex_plot)
```

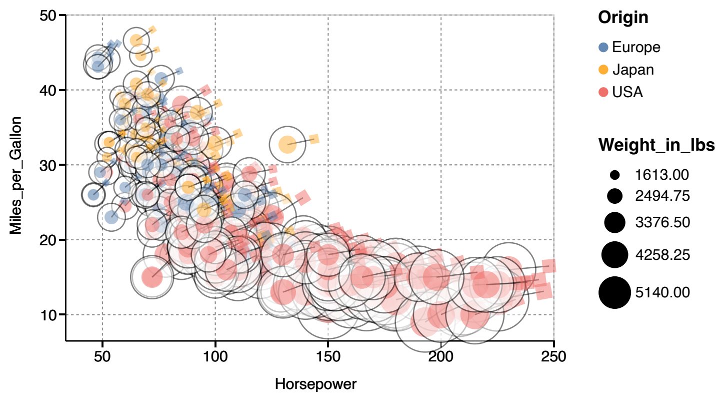

Figure 3: Custom scatter plot created with a graphic expression.

Lastly, we illustrate how to combine plots using diagramming operations. Since a plot is a mark, it can be manipulated as any other mark. The following example shows how to combine the two plots created in the previous examples:

```
viz_title = TextMark(text="Showcasing Diagramming Integration",anchor=:c,fontsize=20)
viz_frame = S(:fillOpacity=>0,:stroke=>:black)*T(400,100)*Rectangle(h=370,w=1000)

new_viz = simple_plot + T(470,0)*complex_plot + viz_frame + T(400,250)*viz_title

draw(new_viz)
```

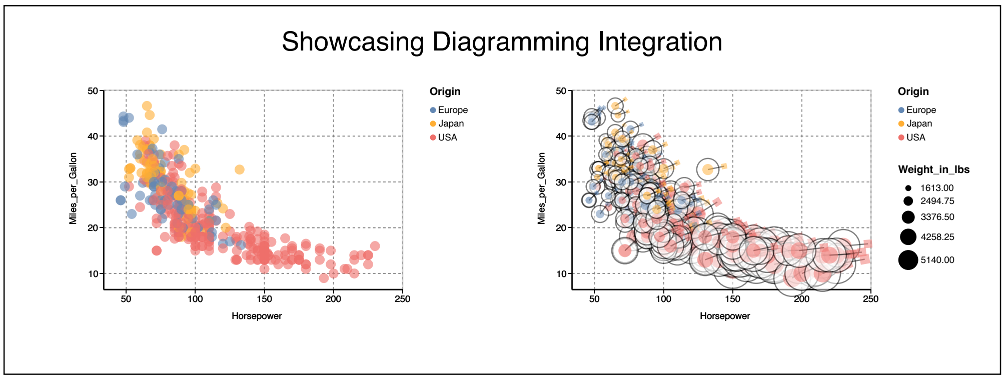

Figure 4: Combining plots using diagramming operations.

A more detailed overview of the functionality of Vizagrams.jl is provided in the documentation, where it goes into details about the diagramming DSL, the mark creation process, the visualization grammar, and the use of graphic expressions.

References

Barreira, D. S. (2025). *Data visualization from a category theory perspective* [PhD thesis, Fundação Getulio Vargas]. https://hdl.handle.net/10438/36341

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, *17*(12), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

Bouchet-Valat, M., & Kamiński, B. (2023). Dataframes.jl: Flexible and fast tabular data in Julia. *Journal of Statistical Software*, *107*, 1–32. https://doi.org/10.18637/jss.v107.i04

Breloff, T. (2025). *Plots.jl* (Version v1.40.17). Zenodo. https://doi.org/10.5281/zenodo.15886032

Brunson, J. C. (2020). Ggalluvial: Layered grammar for alluvial plots. *Journal of Open Source Software*, *5*(49), 2017. https://doi.org/10.21105/joss.02017

Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, *6*(65), 3349. https://doi.org/10.21105/joss.03349

Giovine, S. (2022). *Compose.jl: Declarative vector graphics for Julia*. https://github.com/GiovineItalia/Compose.jl

Granström, P. (2022). *Diagrammar: Simply make interactive diagrams*. Strange Loop;

Talk.https://www.thestrangeloop.com/2022/diagrammar-simply-make-interactive-diagrams.html

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Jones, D. C., Arthur, B., Nagy, T., Mattriks, Gowda, S., Godisemo, Holy, T., Noack, A., Sengupta, A., Darakananda, D., B, A., Dunning, I., Leblanc, S., Huijzer, R., Fischer, K., Chudzicki, D., Piibeleht, M., Mellnik, A., Kleinschmidt, D., … Saba, E. (2021). *GiovineItalia/gadfly.jl: v1.3.4* (Version v1.3.4). Zenodo. https://doi.org/10.5281/zenodo.5559613

Li, G., Tian, M., Xu, Q., McGuffin, M. J., & Yuan, X. (2020). Gotree: A grammar of tree visualizations. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3313831.3376297

Liu, Z., Chen, C., Morales, F., & Zhao, Y. (2021). Atlas: Grammar-based procedural generation of data visualizations. *2021 IEEE Visualization Conference (VIS)*, 171–175. https://doi.org/10.1109/vis49827.2021.9623315

Liu, Z., Thompson, J., Wilson, A., Dontcheva, M., Delorey, J., Grigg, S., Kerr, B., & Stasko, J. (2018). Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3173574.3173697

LY'i, S., Wang, Q., Lekschas, F., & Gehlenborg, N. (2021). Gosling: A grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Transactions on Visualization and Computer Graphics*, *28*(1), 140–150. https://doi.org/10.1109/tvcg.2021.3114876

Mei, H., Ma, Y., Wei, Y., & Chen, W. (2018). The design space of construction tools for information visualization: A survey. *Journal of Visual Languages & Computing*, *44*, 120–132. https://doi.org/10.1016/j.jvlc.2017.10.001

Morel, P. (2018). Gramm: Grammar of graphics plotting in Matlab. *Journal of Open Source Software*, *3*(23), 568. https://doi.org/10.21105/joss.00568

Observable plot. (2024). https://github.com/observablehq/plot.

Pollock, J., Mei, C., Huang, G., Evans, E., Jackson, D., & Satyanarayan, A. (2024). Bluefish: Composing diagrams with declarative relations. *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 1–21. https://doi.org/10.1145/3654777.3676465

Pu, X., & Kay, M. (2020). A probabilistic grammar of graphics. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3313831.3376466

Ren, D., Lee, B., & Brehmer, M. (2018). Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, *25*(1), 789–799. https://doi.org/10.1109/tvcg.2018.2865158

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. (2014). *Vega: A visualization grammar*. http://trifacta.github.io/vega.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. (2016). Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 341–350. https://doi.org/10.1109/TVCG.2016.2599030

Stolte, C., Tang, D., & 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. https://doi.org/10.1109/2945.981851

Tsandilas, T. (2020). StructGraphics: Flexible visualization design through data-agnostic and reusable graphical structures. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 315–325. https://doi.org/10.1109/tvcg.2020.3030476

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Weaver, C. (2004). Building highly-coordinated visualizations in improvise. *IEEE Symposium on Information Visualization*, 159–166. https://doi.org/10.1109/infvis.2004.12

Wickham, H. (2011a). ggplot2. *Wiley Interdisciplinary Reviews: Computational Statistics*, *3*(2), 180–185. https://doi.org/10.1002/wics.147

Wickham, H. (2011b). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, *40*, 1–29. https://doi.org/10.18637/jss.v040.i01

Wickham, H., & Wickham, M. H. (2007). The ggplot package. In *URL: https://cran.r-project.org/web/packages/ggplot2/index.html*.

Wilkinson, L. (2012). The grammar of graphics. In *Handbook of computational statistics* (pp. 375–414). Springer. https://doi.org/10.1007/978-3-642-21551-3_13

Yates, R., & Yorgey, B. A. (2015). Diagrams: A functional EDSL for vector graphics. *Proceedings of the 3rd ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design*, 4–5. https://doi.org/10.1145/2808083.2808085

Ye, K., Ni, W., Krieger, M., Ma'ayan, D., Wise, J., Aldrich, J., Sunshine, J., & Crane, K. (2020). Penrose: From mathematical notation to beautiful diagrams. *ACM Transactions on Graphics (TOG)*, *39*(4), 144–141. https://doi.org/10.1145/3386569.3392375

Yorgey, B. A. (2012). Monoids: Theme and variations (functional pearl). *ACM SIGPLAN Notices*, *47*(12), 105–116. https://doi.org/10.1145/2430532.2364520