The Journal of Open Source Software

grplot: A Python Library for Lazy Statistical Data Visualization

Ghiffary Rifqialdi [1,2,3¶]

1 University of Hamburg, Hamburg, Germany **2** University of L'Aquila, L'Aquila, Italy **3** Institut Teknologi Bandung, Bandung, Indonesia ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 25 February 2026
Published: unpublished

Summary

`grplot` is an open-source Python library that reduces multi-step statistical plotting workflows to a single high-level function call. Built on top of Matplotlib (Hunter, 2007), NumPy (Harris et al., 2020), and Pandas (McKinney, 2010), and bundled with a vendored Seaborn fork (`grplot_seaborn`) that internally uses SciPy (Virtanen et al., 2020), it exposes a unified `plot2d` API that automatically handles subplot layout, axis labeling, legends, statistical annotations, tick-label number formatting (thousand separators, currency symbols, and magnitude abbreviations), and figure export to PNG, PDF, SVG, and EPS. Users specify what to plot via a plot-type string or a dictionary mapping panel positions to plot types; the library applies sensible defaults while accepting more than 100 parameters for explicit overrides at global, per-axis, or per-element granularity. As of version 1.0.6, `grplot` requires Python 3.10+ and is available on PyPI (v1.0.6) and conda-forge (currently distributed as v1.0.4; the conda-forge feedstock lags PyPI releases) (Rifqialdi, 2026). Full documentation is available at grplot.readthedocs.io.

Statement of Need

Producing publication-quality figures in Python typically requires orchestrating several libraries: constructing Figure/Axes objects in Matplotlib, calling Seaborn chart functions, manually setting tick formats, adding text annotations, adjusting legends, and finally saving the output. For practitioners who generate many plots routinely—data analysts, researchers, and data scientists—this boilerplate is repetitive and error-prone.

`grplot` fills this gap with a consistent imperative API. It is particularly suited to data practitioners who need reproducible, annotated figures in notebooks or technical reports without rebuilding formatting utilities for each project. It also ships two domain-specific analytic utilities—cohort retention analysis and rank-order/gain/KS/lift tables—that practitioners commonly need to reimplement from scratch.

State of the Field

Several Python libraries address statistical data visualization, each with a different scope. Matplotlib (Hunter, 2007) provides a complete 2-D graphics environment with full control over every element, but requires verbose, procedural code for even routine plots. Seaborn (Waskom, 2021) raises the abstraction level for common statistical charts while remaining tightly coupled to Matplotlib's axis-management model. Altair (VanderPlas et al., 2019) and `plotnine` (Kibirige & others, 2022) implement declarative grammars of graphics (Wickham, 2016) that are elegant for exploratory work but do not natively support multi-panel layout, number formatting, or annotation in a single call. Plotly (Plotly Technologies Inc., 2015) excels at interactive web-based visualization but is not oriented toward static, publication-ready figures.

grplot was built rather than contributing to existing projects for three reasons. First, none of the tools above offers a single end-to-end call that combines multi-panel subplot layout, chart rendering, number formatting, inset statistical summaries, value-label annotations, and figure export. Second, the target workflow—generating many annotated figures for notebooks and technical reports—prioritizes brevity and consistency over the full configurability of Matplotlib or the declarative grammar of Altair. Third, the bundled domain-specific analytics (cohort and rank_order) are not available in any of the packages above and would otherwise require separate, custom implementations for each project.

Software Design

Hierarchical Argument System

The central design challenge was exposing a large surface area of configuration (20 chart types, multi-panel grids, per-axis formatting, per-element overrides) through a single function without requiring users to understand its full breadth for routine use. grplot resolves this with a four-level hierarchy of argument granularity:

- **Ordinary**: applied to the entire figure (e.g., `df`, `figsize`, `Nx`).
- **Axes**: scoped to a specific subplot by 1-based index `"[i]"` (1-D) or `"[row,col]"` (2-D grid), e.g., `plot`, `filter`, `title`.
- **Axes-plot**: scoped to a specific chart layer within a subplot (e.g., `hue={"[1,2]": {"scatterplot": "species"}}`).
- **Axes-axislabel**: scoped to a specific axis label within a subplot (e.g., `statdesc={"[1,1]": {"total_bill": "general"}}`).

Almost all axes-axislabel arguments apply to both axes by default; prefixing with `x` or `y` targets a single axis (e.g., `xlim`, `yrot`). This design deliberately trades away the lowest-level Matplotlib configurability in exchange for allowing a complete, multi-panel, annotated figure to be expressed in one call with consistent, predictable defaults. Full parameter documentation is available in the online documentation.

Supported Chart Types

grplot wraps 20 chart types across four families:

Family	Chart types
Relational	scatterplot, lineplot
Distribution	histplot, kdeplot, ecdfplot, rugplot, pieplot, treemapsplot, packedbubblesplot
Categorical	stripplot, swarmplot, boxplot, violinplot, boxenplot, pointplot, barplot, countplot, paretoplot
Regression	regplot, residplot

treemapsplot bundles an inline implementation of the squarified treemap layout algorithm described by Laserson (2013), requiring no external squarify dependency. Any two chart types may be overlaid on the same axis using + notation (e.g., `"histplot+kdeplot"`). Five composites carry pre-tuned default values: `boxplot+stripplot`, `violinplot+stripplot`, `boxplot+swarmplot`, `violinplot+swarmplot`, and `stripplot+pointplot`. Multiple panels can be composed into grid dashboards using `Nx` (columns) and `Ny` (rows), as illustrated in Figure 1.

Vendored Seaborn Fork

`grplot` ships a vendored fork of Seaborn (`grplot_seaborn`) to decouple production software that embeds `grplot` from upstream Seaborn breaking changes. This is a deliberate stability trade-off: users gain version-independence at the cost of not automatically inheriting Seaborn upstream improvements. The fork is kept up to date with stable Seaborn releases as part of `grplot` maintenance.

Analytic Utilities

`grplot.analytic.cohort` produces a cohort retention heatmap from a customer transaction table, computing monthly retention rates in a single call (Figure 2). When `display_summary=True`, the underlying cohort pivot table is also printed to the notebook output for inspection.

`grplot.analytic.rank_order` produces a rank-order table with cumulative gain, KS statistic, and lift per decile from predicted probabilities and true binary labels, supporting multi-class outputs via a class selector. These utilities follow standard industry conventions and remove a common source of bespoke, error-prone reimplementation in data science notebooks.

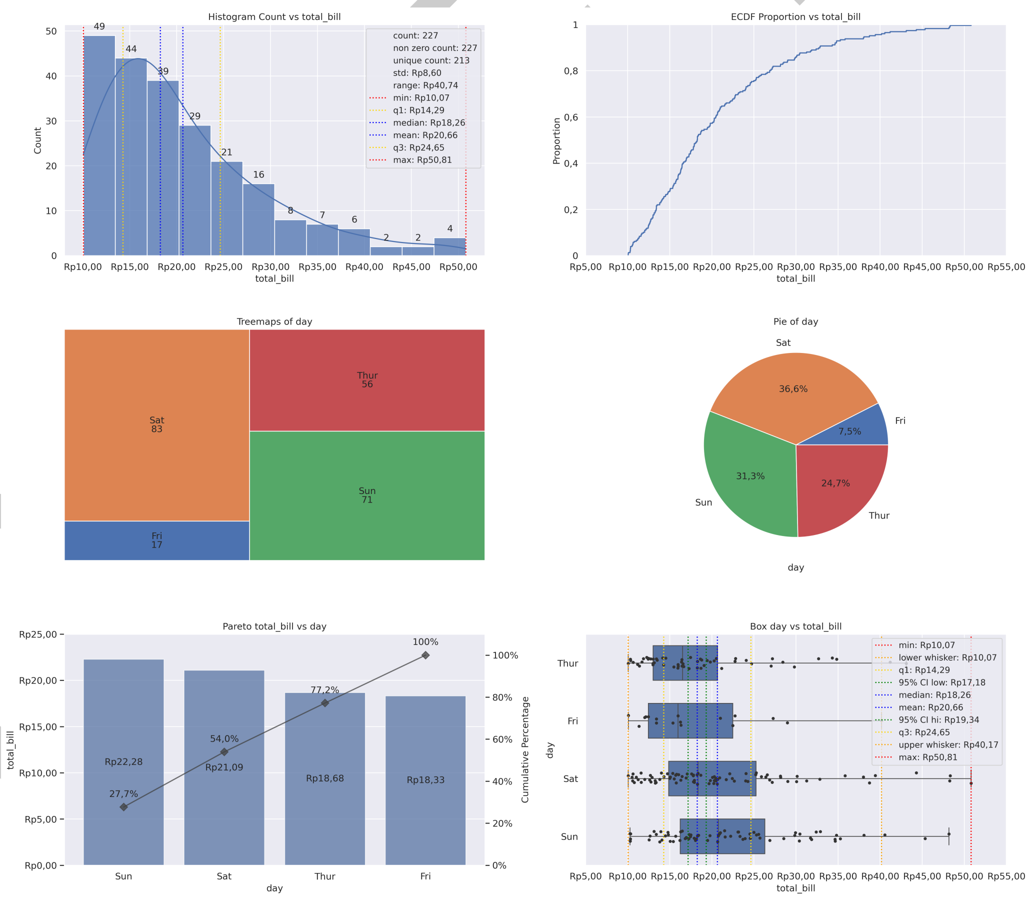

Figure 1: Six-panel 2×3 grid dashboard—histogram, ECDF, treemap, pie, Pareto, and box+strip composite—generated with a single `plot2d` call and a per-panel row filter applied to the Seaborn `tips` dataset. Tick formatting (`Rp(_)`), inset statistical annotation blocks, and bar-top value labels are all configured through `plot2d` parameters without any post-processing.

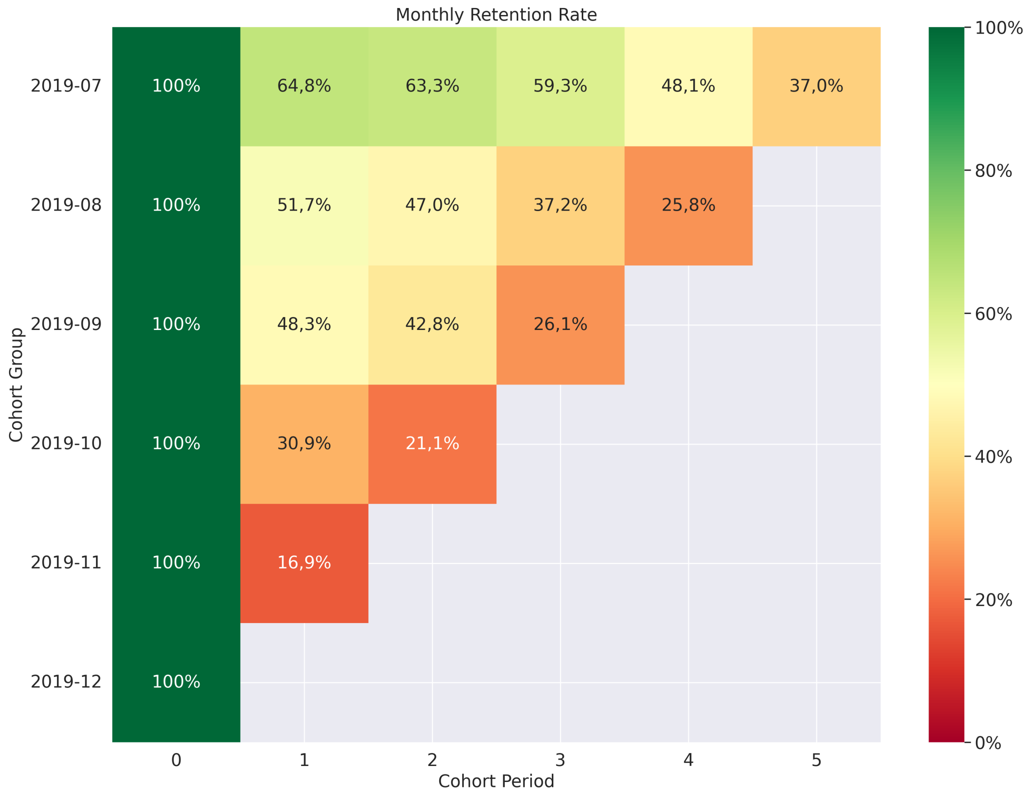

Figure 2: Monthly cohort retention heatmap produced by `grplot.analytic.cohort` from a retail transaction dataset. Rows represent cohort groups (signup month); columns represent cohort periods (months since first purchase); cell values show the percentage of customers active in each subsequent month.

Research Impact Statement

`grplot` reduces the time and code required to produce annotated, publication-ready figures in Python. By consolidating multi-step Matplotlib/Seaborn workflows into a single `plot2d` call with a hierarchical parameter system, it lowers the barrier to exploring and communicating data for data analysts, researchers, and data scientists who may not have deep expertise in lower-level graphics APIs. The bundled analytic utilities (`cohort` and `rank_order`) further accelerate common modeling-evaluation and customer-analysis workflows that practitioners would otherwise rebuild from scratch. `grplot` supports reproducibility by making figure-generation code concise, readable, and easy to version-control, and it integrates naturally into Jupyter notebook environments widely used in data science research.

Since its public release, `grplot` has accumulated more than 98,000 total downloads on PyPI (source: pepy.tech, retrieved 2026-02-28), ranking in the top 10% of packaged Python projects by download volume (source: ClickHouse ClickPy, retrieved 2026-02-28). An interactive Colab documentation notebook serves as a community-readiness signal: it allows practitioners to run all examples in a zero-install environment, and its existence reflects requests from potential users for a lower-friction entry point than a local installation.

AI Usage Disclosure

An AI assistant was used solely to assist with brainstorming and idea development during the writing of this paper. The tool was not used in software creation, code generation, or documentation writing. All technical content, design decisions, and architectural choices are the work of the author alone. All AI-assisted ideation was reviewed, evaluated, and validated by the author before inclusion.

Acknowledgements

The author thanks the maintainers of Matplotlib (Hunter, 2007), Seaborn (Waskom, 2021), NumPy (Harris et al., 2020), Pandas (McKinney, 2010), SciPy (Virtanen et al., 2020), and IPython (Pérez & Granger, 2007) for providing the foundational infrastructure on which `grplot` is built. No financial support was received for this work. The author declares no conflict of interest.

References

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., & others. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

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

Kibirige, H., & others. (2022). *Plotnine: A grammar of graphics for Python*. https://doi.org/10.5281/zenodo.1325308

Laserson, U. (2013). *Squarify: Pure Python implementation of the squarify treemap layout algorithm*. https://github.com/laserson/squarify

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 51–56. https://doi.org/10.25080/Majora-92bf1922-00a

Pérez, F., & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, *9*(3), 21–29. https://doi.org/10.1109/MCSE.2007.53

Plotly Technologies Inc. (2015). *Collaborative data science*. https://plot.ly

Rifqialdi, G. (2026). *Grplot: A python library for lazy statistical data visualization* (Version 1.0.6). https://github.com/ghiffaryr/grplot

VanderPlas, J., Granger, B. E., Heer, J., Moritz, D., Wongsuphasawat, K., Satyanarayan, A., Lees, E., Timofeev, I., Welsh, B., & Sievert, S. (2019). Altair: Interactive statistical visualizations for Python. *Proceedings of the 18th Python in Science Conference*, 98–105. https://doi.org/10.25080/Majora-7ddc1dd1-00f

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., & others. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

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

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer. https://doi.org/10.1007/978-3-319-24277-4