Received: 16 June 2022 | Revised: 14 October 2022 | Accepted: 30 October 2022

DOI: 10.1002/nbm.4865

SPECIAL ISSUE RESEARCH ARTICLE

Visualizing metabolomics data with R

Yannick Berker[1,2,3] | **Isabella H. Muti**[4] | **Leo L. Cheng**[4]

[1]Hopp Children's Cancer Center Heidelberg (KiTZ), Germany
[2]Clinical Cooperation Unit Pediatric Oncology, German Cancer Research Center (DKFZ) and German Cancer Consortium (DKTK), Germany
[3]National Center for Tumor Diseases (NCT) Heidelberg, Germany
[4]Departments of Radiology and Pathology, Harvard Medical School, Massachusetts General Hospital, MA, USA

Correspondence
Yannick Berker, Clinical Cooperation Unit Pediatric Oncology, German Cancer Research Center (DKFZ), Im Neuenheimer Feld 280, 69120 Heidelberg, Germany.
Email: yannick.berker@alumni.dkfz.de

Present Address
Yannick Berker, Siemens Healthineers, Molecular Imaging, 91301 Forchheim, Germany.

In communicating scientific results, convincing data visualization is of utmost importance. Especially in metabolomics, results based on large numbers of dimensions and variables necessitate particular attention in order to convey their message unambiguously to the reader; and in the era of open science, traceability and reproducibility are becoming increasingly important. This paper describes the use of the R programming language to visualize published metabolomics data resulting from ex-vivo NMR spectroscopy and mass spectrometry experiments with a special focus on reproducibility, including example figures as well as associated R code for ease of reuse. Examples include various types of plots (bar plots, swarm plots, and violin plots; volcano plots, heatmaps, Euler diagrams, Kaplan-Meier survival plots) and annotations (groupings, intra-group line connections, significance brackets, text annotations). Advantages of code-generated plots as well as advanced techniques and best practices are discussed.

KEYWORDS:
data visualization, metabolomics, R

1 | INTRODUCTION

Visualization of experimental data is an important issue in any data science. In metabolomics, however, we meet specific challenges due to the high dimensionality of the data: in a usual experiment, hundreds of metabolites each are quantified for dozens, hundreds, or even thousands of samples and analyzed using various statistical methods, for groups and subgroups. As a result, the generation of representations that are accurate, convincing, visually appealing, and consistent is a complicated endeavor.

There is no shortage of tools to generate almost any kind of plot, ranging from general-purpose office-suite applications (Microsoft Excel, Apple Numbers, etc.) to statistics software suites (SAS, JMP, etc.), to dedicated plotting libraries often programmable via high-level programming languages (Veusz; `Matplotlib`, Scikit-plot, and `plotnine` for Python; `ggplot2` for R; MATLAB; etc.). In this tutorial paper, we showcase the use of the R programming language for data visualization with a focus on metabolomics applications. We chose R, for it is open source and freely available, and with `ggplot2`,[1] it offers arguably the most mature implementation of The Grammar of Graphics,[2] enabling the separation of data to be plotted from the plot semantics to a high degree. We will establish necessary terminology through several examples and present best practices to generate and compose consistent, high-quality, publication-ready figure files ready for use in posters, presentations, and journal papers without any postprocessing.

All code used to generate and save the plots in this paper is available for reuse at https://github.com/yannickberker/met-vis-R. Functions saving plots to figure files can be combined with almost any other type of plot: readers may take further inspiration from websites such as The R Graph Gallery.[3]

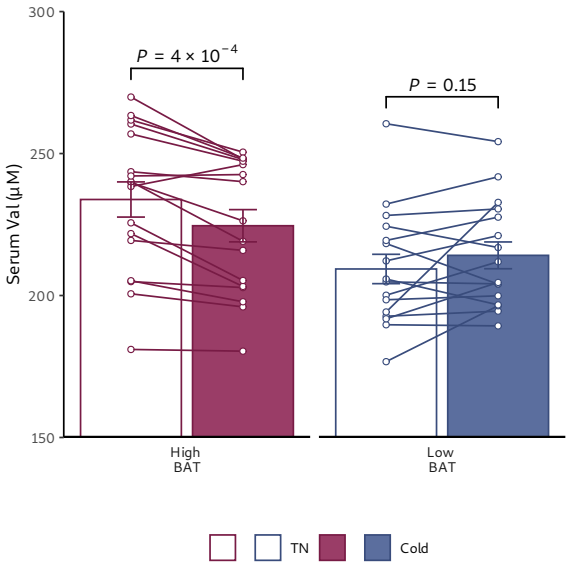

Figure 1 Bar plot using data from Yoneshiro et al.,[5] figure 1(b)

1.1 | What this tutorial does not cover

This tutorial focuses on an introduction into technical aspects of data visualization using examples inspired by metabolomics findings—it does not intend to cover (metabolomic) data analysis in any way, such as optimal ways of dimensionality reduction, data stratification and aggregation, choosing appropriate statistical tests, correcting for multiple testing, or identifying outliers. We will assume that any data analysis has been implemented and carried out and, while later amendments of the data are possible, that the structure of the data to be visualized is determined and the essential data is given in one data file per plot. For simplicity, we will use data stored in Microsoft Excel-compatible files for maximum user convenience while maintaining portability.

While the included code examples may be helpful as a start, implementation details are not generally a topic of this tutorial. Users should refer to the link collection titled *Big Book of R*, specifically chapter 12 (Data Visualization),[4] and ultimately to the reference manuals of R and its respective packages. These will also be helpful in implementing the ideas for advanced data representations put forth in section 4.

Using R code for data visualization does not necessarily contradict using an interactive user interface: users less experienced in writing (R) code may prefer generating basic plots using a `Shiny`-based, interactive user interface offered by the `esquisse` package, which allows exporting of not only the generated plot, but also the code to reproduce the code non-interactively later. While this tutorial does not cover this use, the basic recommendations apply equally to the code thus generated.

2 | GENERATING DIFFERENT TYPES OF PLOTS IN R

The following examples reproduce plots from the published literature, and we are indebted to the authors of the respective publications for their most helpful cooperation. The plots cover various aspects of data visualization, in particular: types of plots (bar plots, swarm plots, and violin plots; volcano plots, heatmaps, Euer diagrams, Kaplan-Meier survival plots) and annotations (groupings, intra-group line connections, significance brackets, text annotations, colors, etc.).

2.1 | Bar plot

Bar plots, also known as bar charts/graphs, are used to visualize categorical data (see Figure 1). Categories are typically plotted on the horizontal axis, and the heights or lengths of the bars correspond to their respective values for the given measured variable. To assist readers, procedures for using the code at https://github.com/yannickberker/met-vis-R to generate the plots in this article are illustrated there using the example of this bar plot and can be applied to the others presented in this article.

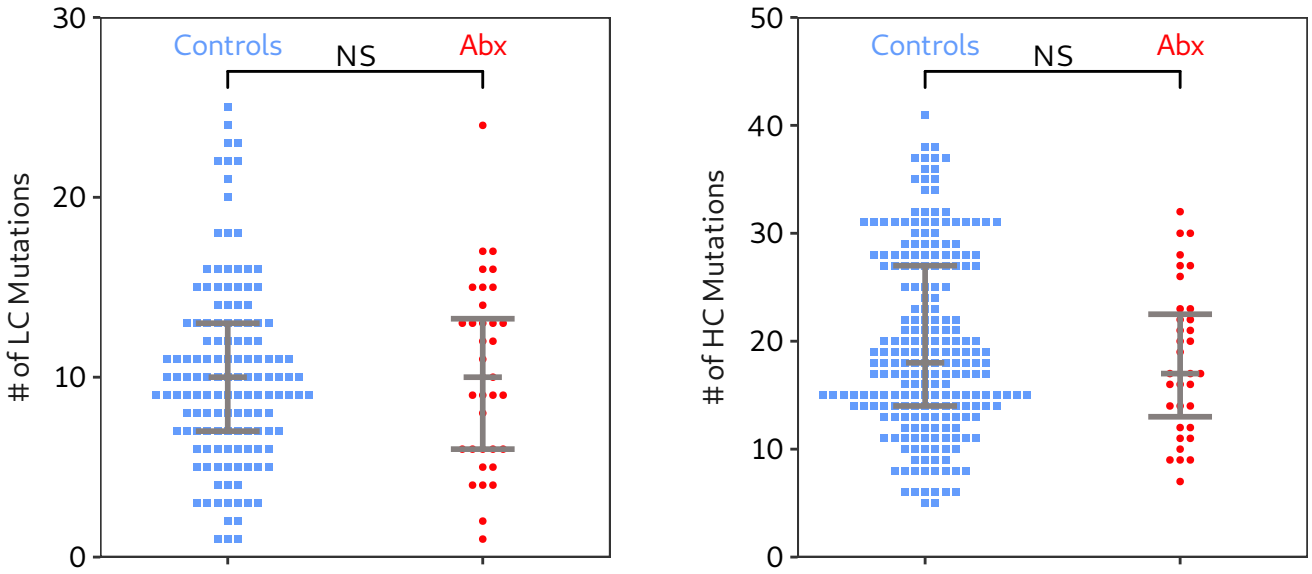

Figure 2 Swarm plot using data from Hagan et al.,[6] figure S3

2.2 | Swarm plot

Swarm plots show the distribution of a continuous numerical attribute across another, categorical attribute. Swarm plots include all data points, so they are best used for relatively small data sets. See Figure 2.

2.3 | Violin plots

Similar to swarm plots, violin plots are used to plot continuous numerical data across different categories. They are similar to box plots in that they show the distribution of the data, but they also include the density of each variable and therefore show peaks in the data. The wider sections of the violin plot show areas of higher data point density while the narrower sections represent a lower probability of having a data point. See Figure 3 for an overlaid comparison of violin and swarm plots.

2.3.1 | Volcano plot

Scatter plots are commonly used to plot two paired continuous numerical attributes, and used to visualize correlations. They are also ideally suited to visualize the result of dimensionality reduction techniques such as principal component analysis, t-SNE,[7] or UMAP.[8]

A volcano plot is a special scatter plot that depicts significance (e.g., p value) versus magnitude of change (e.g. fold change) from a statistical test. Volcano plots are useful in that they show large changes within the data set that have statistical significance. The negative logarithm of the p value is typically plotted on the vertical (y-)axis, and the horizontal (x-)axis depicts the logarithm of the fold change. As a result, more significant data points with lower p values are represented at the top of the plot. Data points on the far left and right of the plot are also of interest, as they represent points with large fold changes. See Figure 4.

2.4 | Heatmap

Heatmaps are two-dimensional plots that represent the magnitude of a data association using differences in color and intensity. In a typical cluster heatmap, data associations are visualized as a matrix of cells, with each row and column representing a category, and each cell is shaded with a color and intensity that corresponds to an association level. See Figure 5.

Heatmaps provide a unique opportunity to visualize the results of correlation analyses. In a typical setting, a *correlation heatmap* is calculated by computing correlation coefficients between pairs of attributes A and B along all samples; correlation strengths are typically color-coded. Such a correlation heatmap is typically symmetric as correlation coefficients do not consider the order of their arguments, and each half of such a correlation heatmap carries the full amount of information.

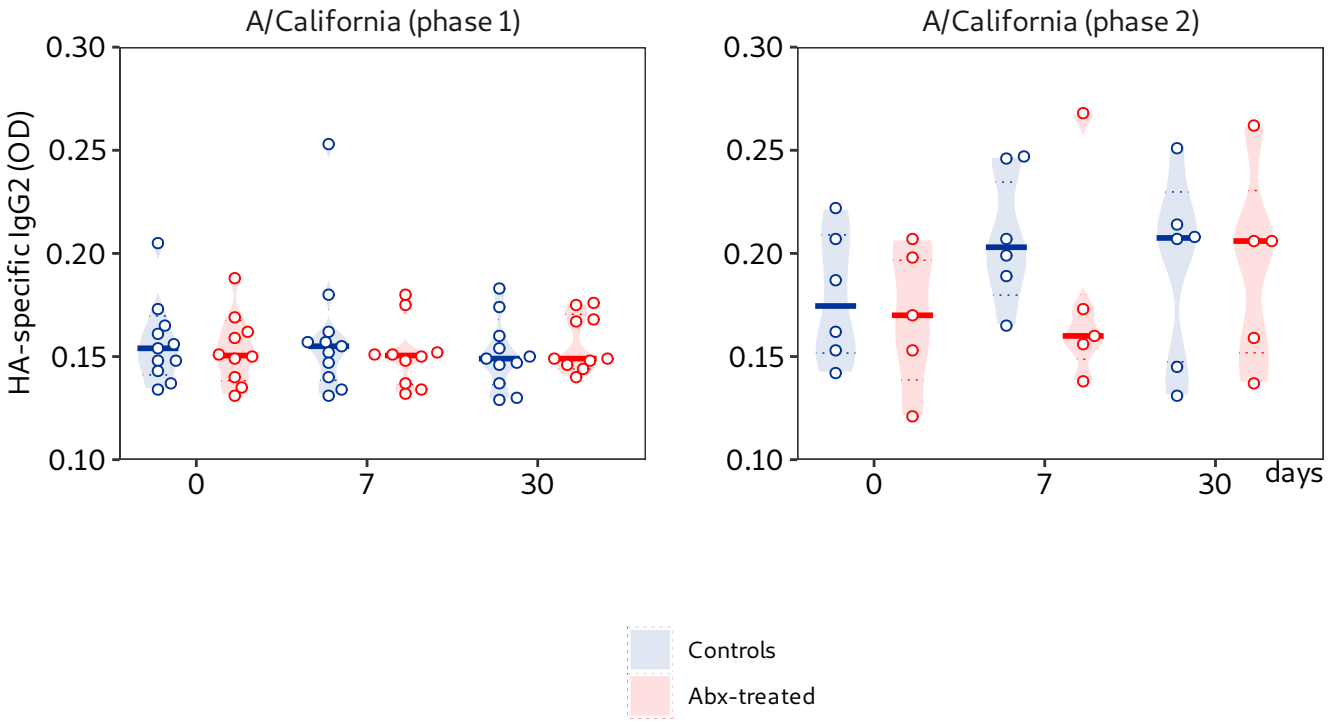

Figure 3 Violin/swarm plot using data from Hagan et al.,[6] figure S2C

This concept can be extended in metabolomics, where paired samples from the same subject (e.g., serum and tissue) are often studied. The aforementioned symmetry can be broken by computing correlation coefficient across samples, namely, correlating attribute A from tissue samples with attribute B from serum samples, and vice versa, providing an additional correlation-heatmap and thus, an additional layer of visualization.

2.5 | Euler diagrams

A Euler diagram is used to represent the relationships between sets of data. They are similar to Venn diagrams in that they have overlapping regions, which show the shared characteristics between sets. They are distinct from Venn diagrams, however, in that they do not have to show an entire set. They are thus useful for representing complex data sets in a simplified manner. See Figure 6.

Of special note, comparing with Shen et al.,[11] figure S4 our current plot reflects both the relationships and area proportions between subgroups in the plots, while the original plots are only illustrative of the area proportions between groups.

2.6 | Kaplan-Meier plot

The Kaplan-Meier plot is used to represent the estimated survival function from a data set. In practice, it is often used in medical research to show the proportion of patients who survive a certain amount of time with a disease with or without progression. The plot is visualized as a curve of declining horizontal steps that approach the survival function. See Figure 7.

3 | COMBINING PLOTS

Visual representations often benefit from juxtaposition of related information, to emphasize contrast or similarity. As such, *grids* are a way of automatically combining subplots, giving the authors great control over placement and arrangement of multiple plots that will usually survive the copyediting process. See Figure 8 for an example combining three of the previous figures; notably, the grid code reuses the code of the individual figures, avoiding code duplication and hence potential inconsistencies; and each individual figure is laid out according to the new geometry without

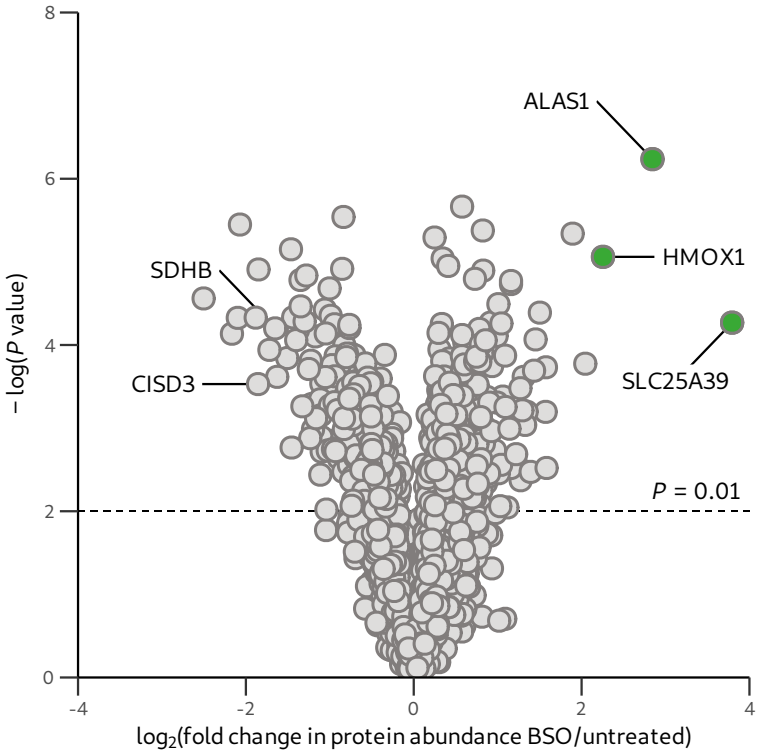

Figure 4 Volcano plot using data from Wang et al.,[9] figure 1c

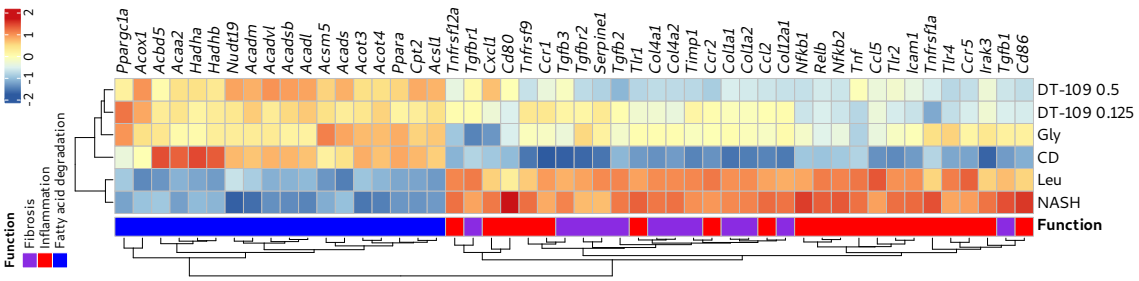

Figure 5 Heatmap using data from Rom et al.,[10] figure 6

stretching or squeezing. Similarly, Figure 6 sets two plots side by side using the `cowplot` package and Figures 2 and 3 demonstrate the concept of `ggplot2` *facets* (compare with the following section).

4 | ADVANCED DATA REPRESENTATIONS

In addition to the example plots showcased above, which usually direct the reader's attention towards a single aspect of the underlying data, several advanced techniques shall also be mentioned.

As shown above, grids are one way to produce pixel-perfect arrangements and compositions of subfigures in one figure panel within R, skipping the time-consuming manual composition (which is also error-prone, especially in case of updates to subplots). Using similar techniques, *inserts* can be created, effectively embedding a smaller plot in the unused space of another.

Automating the concept of grids, facets (in `ggplot2` terminology) are a way of repeating a plot for several to many subgroups of the data, as indicated in Figures 2 and 3 for two subgroups, respectively. While this use of multifaceted figures seems to be rare in some journals where space

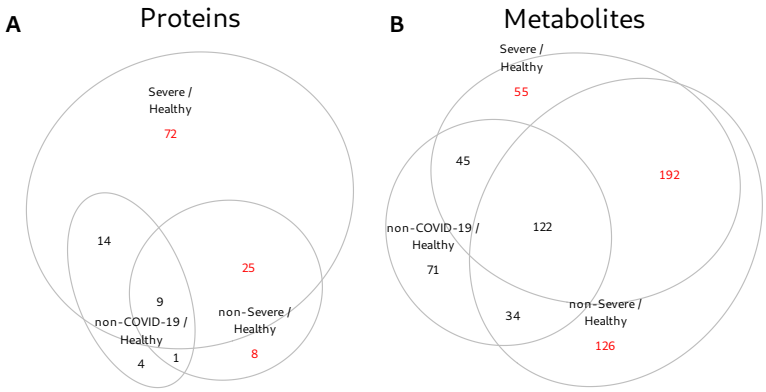

Figure 6 Euler diagram using data from Shen et al.,[11] figure S4

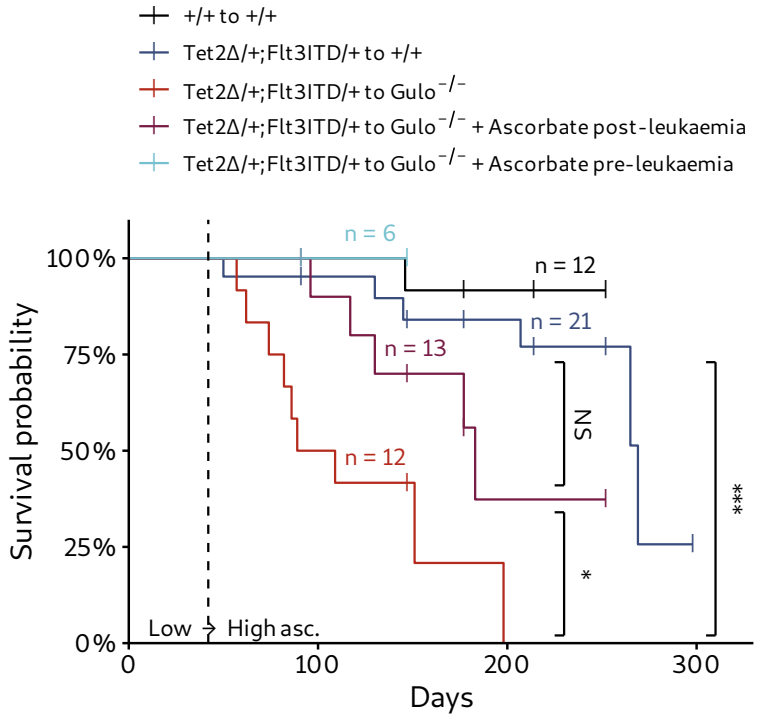

Figure 7 Kaplan-Meier plot using data from Agathocleous et al.,[12] figure 5(a,b)

constraints are applied, they are a great way of inspecting longitudinal data by distributing them along one dimension while keeping all other aesthetics (e.g., axes, colors, etc.) consistent.

Finally, *interactive* and *dynamic* plots can be created to allow the reader to interact with larger, multidimensional data, allowing inspection of individual data points in detail or selecting facets on the fly. These kinds of plots are usually hosted online as a supplement to static visualization in a manuscript.

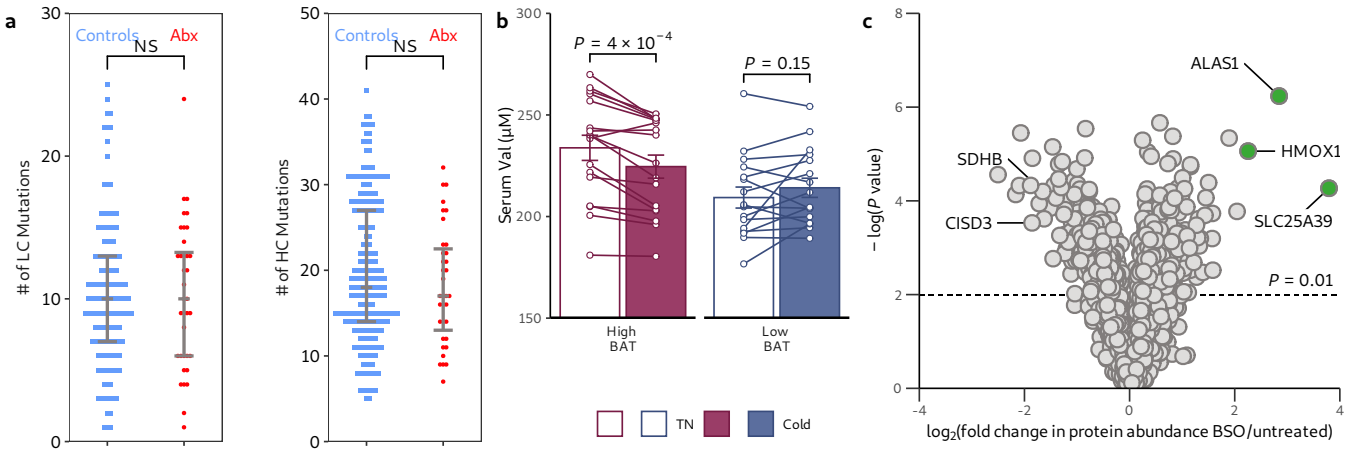

Figure 8 Grid combining the following figures. A, Figure 2. B, Figure 1. C, Figure 4

5 | EXPORTING FIGURES

Exporting figures in the format requested by outlet guidelines (which include the file format, figure dimensions, and figure resolution) can be a struggle in and of itself. Fortunately, R provides a host of graphic devices in the built-in `grDevices` package, natively supporting the export to vector (PDF, PS, SVG) or bitmap (PNG, JPEG, BMP, TIFF) formats. Alternative graphic devices are available via the `Cairo` or `ragg` packages, for instance.

The code provided with this manuscript provides helper functions `(gg)save_cairo_pdf_and_png` to export plots at chosen dimensions and resolution in both PDF and PNG formats, covering the most popular of vector and bitmap formats, respectively.

6 | BEST PRACTICES AND ADVANTAGES

Best practices for using R code to visualize metabolic data rest on the usual best practices for writing computer programming code, including coding style and version control.

Code should follow a style guide – such as in the case of R code, the one put forth by `tidyverse`.[13] Users choosing to adhere to this style are supported by packages such as `styler`[14] to automatically reformat code and `lintr`[15] to perform automated checks of conformance (especially for topics not covered by `lintr`, such as syntactically and logically incorrect code). While the benefit for a single plot author may seem small, other readers of the code – including the author's future self – will be grateful. Remember that *code is written once but read many times*.[†] Along the same lines, code should be commented to ease future reading and reuse, keeping in mind that code comments should not (only) describe *what* the code is doing, but also *why*.

Just like other unformatted text files, `.R` files are useful to keep under version control, for example, using systems such as the de-facto standard `git`(https://git-scm.com/). The decentralized, server-less nature of `git` allows migrating repositories online or offline at any point in time, and `git` repositories can be hosted (with a basic set of attributes) for free at services such as GitHub.com, GitLab.com, and others. In fact, `.R` files to generate the plots in this tutorial are available at https://github.com/yannickberker/met-vis-R. While the learning curve is steep for command-line use of `git`, integrated development environments such as RStudio (https://www.rstudio.com) commonly integrate support for basic `git` operations through a graphical user interface.

Developing R code under version control, if used consistently, means that past versions of plots can be reproduced without keeping a stash of files named `Plot3_v2_final_YB_LLC_final_larger.png` and the like. Plots of identical content can later be re-produced in different variants (compare with section 5), e.g., at different resolutions and file sizes, or to generate landscape bitmaps in PNG format for PowerPoint presentations as well as portrait vector-graphics in PDF format at journal-specific figure dimensions. (Generally, the author saves every plot in both file formats.) If plots are created with all proper annotations within R, supposedly final paper figures can be amended in minutes to resize the resulting figure, or to insert additional data or remove outliers or patient samples after late retraction of consent (god forbid!). This becomes an invaluable time-saver, especially if multi-panel figures are composed within R, removing any need to repeatedly re-compose figures in Acrobat or PowerPoint.

[†] While it is challenging to attribute this statement to one author, the basic idea dates back to at least the 1980s, when "emphasis was placed on program readability over ease of writing" already.[16]

R is extensible by a universe of general-purposed as well as domain-specific packages. For maximum reproducibility, reusability, and security, users should choose packages and package versions that are published in recognized and reviewed repositories such as CRAN (https://cran.r-project.org/) or BioConductor (https://bioconductor.org/).

7 | ALTERNATIVES

R and `ggplot2` may be an excellent basis for a data visualization technology stack, but it is far from being the only choice – for example, if research code in written in Python rather than in R. Interfacing is usually possible by writing data file to disk in commonly-known file formats such as comma-separated values, or by calling R code from Python processes using dedicated packages such as `rpy2` (https://rpy2.github.io/). Users who wish to stay within the Python world may prefer using `matplotlib` (https://matplotlib.org/), which features a MATLAB-inspired application programming interface, or its extension `plotnine` which applies the Grammar-of-Graphics concept to Python (while inheriting several of `matplotlib`'s inherent limitations).

8 | CONCLUSIONS

Given the vastness of the universe of R packages, R presents itself as an ideal choice to generate most, if not all, of today's metabolomics data visualizations, promising a high degree of reproducibility especially if paired with R-based data processing.

Acknowledgements

We sincerely thank the authors for sending us their original data presented in their original papers that allowed us to consider and to reproduce them here using R: Takeshi Yoneshiro and Shingo Kajimura, Thomas Hagan and Bali Pulendran, Kivanç Birsoy, Oren Rom and Y. Eugene Chen, Tiannan Guo, Claudia Langenberg, Noam Bar and Eran Segal. This work was supported in part by the National Institute on Aging of the National Institutes of Health under award number R01AG070257. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. Open Access funding enabled and organized by Projekt DEAL.

Author contributions

Yannick Berker: methodology; data curation; software; writing—original draft preparation; writing—review and editing. **Isabella H. Muti**: validation; writing—original draft preparation; writing—review and editing. **Leo L. Cheng**: conceptualization; funding acquisition; supervision; resources; writing—original draft preparation; writing—review and editing.

ORCID

Yannick Berker https://orcid.org/0000-0002-6707-0834
Isabella H. Muti https://orcid.org/0000-0002-5216-2306
Leo L. Cheng https://orcid.org/0000-0001-5975-5406

References

1. Wickham H. *ggplot2: Elegant Graphics for Data Analysis*. Use R! Cham, Switzerland: Springer. 2nd ed. 2016. doi:10.1007/978-3-319-24277-4
2. Wilkinson L. *The Grammar of Graphics*. Statistics and Computing. New York, NY: Springer. 2nd ed. 2005. doi:10.1007/0-387-28695-0
3. Holtz Y. The R Graph Gallery — Help and inspiration for R charts. https://www.r-graph-gallery.com; 2022. Accessed November 25, 2022.
4. Baruffa O. Big Book of R. https://www.bigbookofr.com/data-visualization.html; 2022. Accessed November 25, 2022.
5. Yoneshiro T, Wang Q, Tajima K, et al. BCAA catabolism in brown fat controls energy homeostasis through SLC25A44. *Nature* 2019; 572(7771): 614-619.

6. Hagan T, Cortese M, Rouphael N, et al. Antibiotics-Driven Gut Microbiome Perturbation Alters Immunity to Vaccines in Humans. *Cell* 2019; 178(6): 1313-1328.e13.
7. van der Maaten L, Hinton G. Visualizing Data using t-SNE. *J Mach Learn Res* 2008; 9(86): 2579-2605.
8. McInnes L, Healy J, Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* 2018: 1802.03426.
9. Wang Y, Yen FS, Zhu XG, et al. SLC25A39 is necessary for mitochondrial glutathione import in mammalian cells. *Nature* 2021; 599(7883): 136-140.
10. Rom O, Liu Y, Liu Z, et al. Glycine-based treatment ameliorates NAFLD by modulating fatty acid oxidation, glutathione synthesis, and the gut microbiome. *Sci Transl Med* 2020; 12(572): eaaz2841.
11. Shen B, Yi X, Sun Y, et al. Proteomic and Metabolomic Characterization of COVID-19 Patient Sera. *Cell* 2020; 182(1): 59-72.e15.
12. Agathocleous M, Meacham CE, Burgess RJ, et al. Ascorbate regulates haematopoietic stem cell function and leukaemogenesis. *Nature* 2017; 549(7673): 476-481.
13. Wickham H. The tidyverse style guide. https://style.tidyverse.org; 2022. Accessed November 25, 2022.
14. Müller K, Walthert L, Patil I. styler: Non-Invasive Pretty Printing of R Code. https://CRAN.R-project.org/package=styler; 2022. Accessed November 25, 2022.
15. Hester J, Angly F, Hyde R, et al. lintr: A 'Linter' for R Code. https://CRAN.R-project.org/package=lintr; 2022. Accessed November 25, 2022.
16. Ada Information Clearinghouse . Ada '83 Language Reference Manual. http://archive.adaic.com/standards/83lrm/html/lrm-01-03.html; 1983. Accessed November 25, 2022.

How to cite this article: Berker Y, Muti IH, Cheng LL. Visualizing metabolomics data with R, *NMR in Biomedicine*, 2022;e4865. doi:10.1002/nbm.4865