frontiers | Frontiers in Genetics

TYPE Technology and Code
PUBLISHED 27 March 2025
DOI 10.3389/fgene.2025.1535368

OPEN ACCESS

EDITED BY
Lei Chen,
Shanghai Maritime University, China

REVIEWED BY
Fabio Iannelli,
European Institute of Oncology (IEO), Italy
Archana Prabahar,
Cleveland State University, United States

*CORRESPONDENCE
Tianze Cao,
tianze-cao@hznu.edu.cn
Yuexia Huang,
yxhuang@hznu.edu.cn

RECEIVED 27 November 2024
ACCEPTED 17 March 2025
PUBLISHED 27 March 2025

CITATION
Zhang Z, Cao T, Huang Y and Xia Y (2025) CirclizePlus: using ggplot2 feature to write readable R code for circular visualization. *Front. Genet.* 16:1535368. doi: 10.3389/fgene.2025.1535368

CirclizePlus: using ggplot2 feature to write readable R code for circular visualization

Zheyu Zhang, Tianze Cao*, Yuexia Huang* and Yu Xia

School of Mathematics, Hangzhou Normal University, Hangzhou, China

In the R programming language, the *de facto* standard framework for drawing rectangular coordinates is ggplot2. The most important feature of ggplot2 is that it is object-oriented and uses the plus sign to overlay various objects. In the field of circular visualization, circlize is a popular software, but it is based on procedural programming. Making it object-oriented can make the logic of the written code clearer and improve the reusability of the code. In this work, we introduce circlizePlus, which redesigns the concepts in circular visualization into several R S4 classes. It also defines a set of additional rules, based on which users can implement ggplot2-like drawing techniques. circlizePlus is a wrapper for circlize. It transforms the procedural programming style of circular visualization drawing into object-oriented programming. The additional rules it defines reduce the amount of coding and make the code more readable. The source codes can be found at https://github.com/tianzelab/circlizePlus, and the sample code can be found at https://tianzelab.github.io/circlizePlusBook/.

KEYWORDS

circlize, ggplot2, object-oriented, generic functions, functional programming

1 Introduction

Circular layout plots typically consist of sectors and tracks. The intersection of sectors and tracks is called a cell (Figure 1). This kind of layout can not only transparently represent the relationship between different data categories, but also rep-resent different observations from different dimensions within the same category. For instance, in genomics, different sectors may represent different chromosomes. In contrast, every single track layer can represent a different data category, such as gene density, or the variation standard, to name but a few.

Although numerous packages have been developed to implement circle layout plotting (Krzywinski et al., 2009; Darzentas, 2010; Zhang et al., 2013; Gu et al., 2014; An et al., 2015; Cheong et al., 2015; Cui et al., 2016; Diaz-Garcia et al., 2017; Drori et al., 2017; Yu et al., 2018; Marx and Coon, 2019; Cui et al., 2020; Rasche and Hiltemann, 2020; Cui et al., 2021; Ennis et al., 2023), circlize (Gu et al., 2014) has become the most mainstream tool since its release. It is an R package that offers various tools and functions to form these complex circle layout plots. Based on the statistical and graphical syntax of R, the package is especially easy for those who are good at R to use. By using circlize package, users can generate various types of charts, including scatter plots, histograms, line plots, heatmaps, etc.

By using circlize package, users can create many sorts of graphical elements easily, such as adding points, lines, texts, and axes. For instance, circos.points () function may be used for adding points, circos.line () for lines, and circos.text () for text labels. Moreover, the

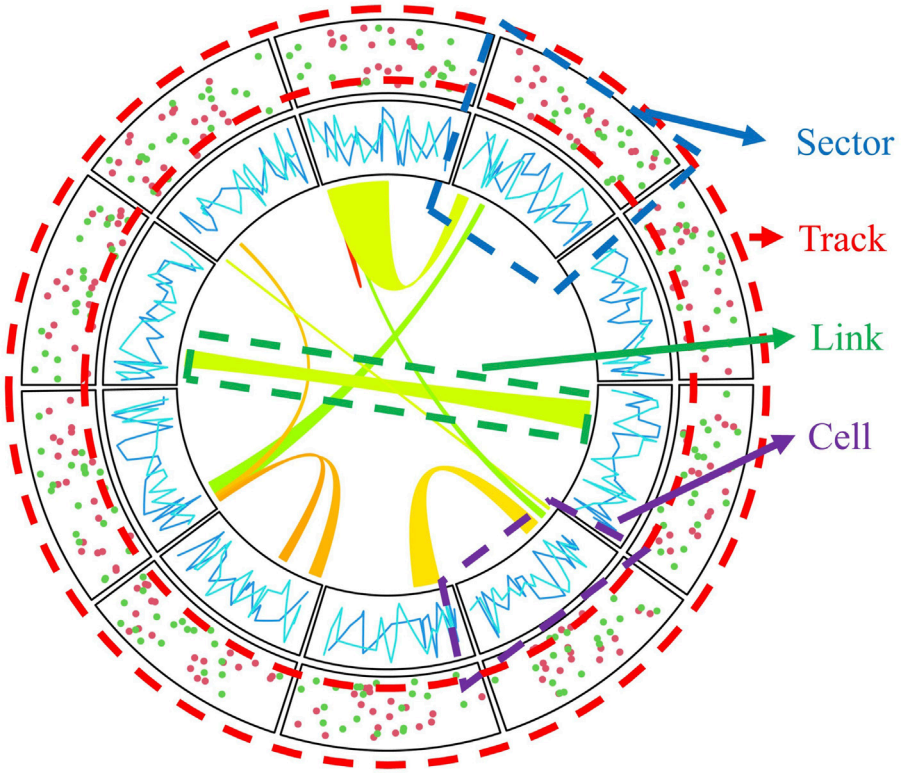

FIGURE 1
Schematic diagram of circle layout plotting.

package is also accessible for stacking tracks and dividing sections, allowing every dataset to be displayed in the appropriate position.

In the domain of rectangular layout plotting, ggplot2 (Wickham, 2016) offers high-level flexibility and controllability by object-oriented programming and the overloading of the addition operator, which makes it much more popular than the basic R plotting API.

However, the classic way to use circlize is procedural programming, so it is necessary to provide support for object-oriented programming and addition operators like ggplot2. This will allow users to overlay different graphic elements in a more intuitive and modular way when creating complex circular layouts. Here, we'd like to introduce circlizePlus, which implements object-oriented programming and additive operations in circular layout plotting by wrapping circlize.

2 Materials and methods

2.1 Classes and addition rules in circlizePlus

The classes and object constructors defined in circlizePlus are prefixed with "cc," to avoid naming conflicts with other packages and represent the abbreviation of circlize. circlizePlus defines 12 R S4 classes, 7 of which are primary classes, namely, ccPlot, ccPar, ccTrack, ccTrackGeom, ccLink, ccCell, and ccCellGeom. Based on these primary classes, circlizePlus constructs a set of addition operation rules as follows.

$$\begin{aligned}
&ccPlot\,(contain\ n\ ccPars) + ccPar \\
&\quad = ccPlot\,(contain\ n+1\ ccPars), n \geq 0 \\
&ccPlot\,(contain\ n\ ccTracks) + ccTrack \\
&\quad = ccPlot\,(contain\ n+1\ ccTracks), n \geq 0 \\
&ccPlot\,(contain\ n\ ccLinks) + ccLink \\
&\quad = ccPlot\,(contain\ n+1\ ccLinks), n \geq 0 \\
&ccTrak\,(contain\ n\ ccTrakGeoms) + ccTrackGeom \\
&\quad = ccTrack\,(contain\ n+1\ ccTrackGeoms), n \geq 0 \\
&ccTrack\,(contain\ n\ ccCells) + ccCell \\
&\quad = ccTrack\,(contain\ n+1\ ccCells), n \geq 0 \\
&ccCell\,(contain\ n\ ccCellGeoms) + ccCellGeom \\
&\quad = ccCell\,(contain\ n+1\ ccCellGeoms), n \geq 0
\end{aligned}$$

2.2 Data mapping from track to geometry

In ggplot2, the parameter data of the function plotting the geometry can be missing. In this case, the value of the parameter with the same name in the function ggplot () will be taken as the default value. Similar features are also implemented in circlizePlus. The coordinate parameters (such as x, y) of the function that draws geometric figures in the sector can be missing. circlizePlus will extract the data of the corresponding

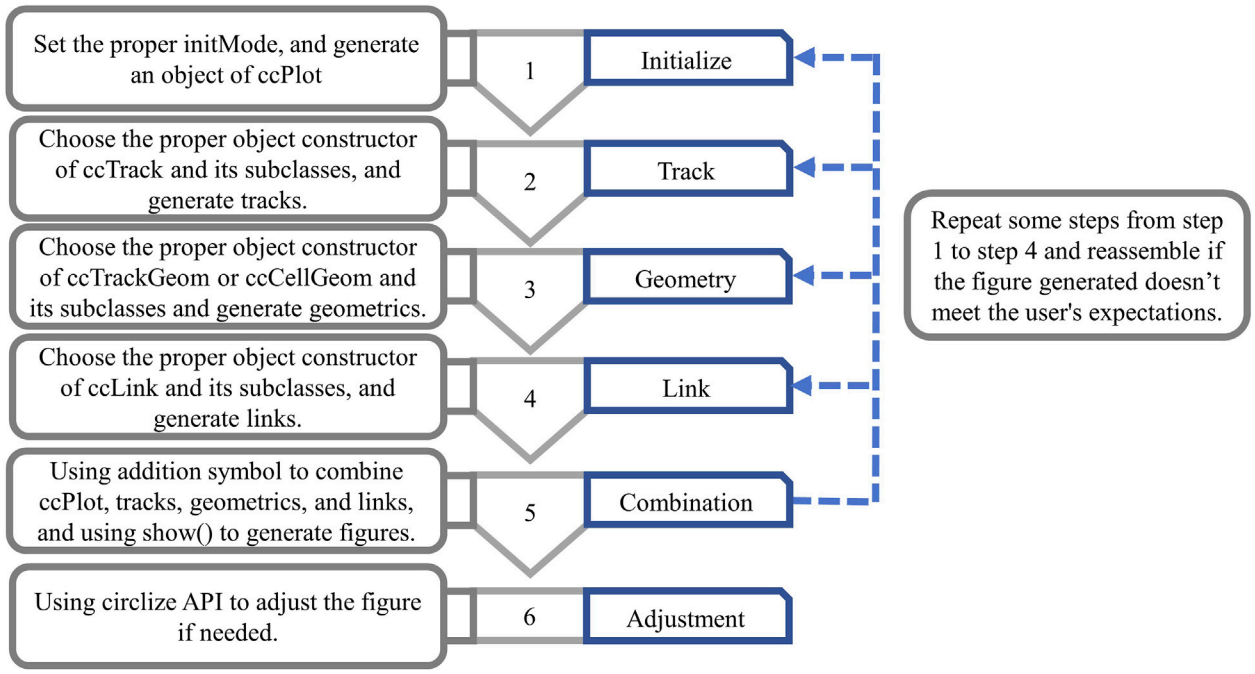

FIGURE 2
Workflow diagram of circlizePlus.

sector of the track it is added to and set it as the default value. It is worth noting that the coordinate parameter can be an anonymous function of the form "function (x, y) . . .". This anonymous function will be called with the default parameter values above, and its re-turn value will be used as the actual value of the coordinate parameter.

2.3 Relationships between ccPar, ccTrack, ccLink and ccPlot

ccPlot is the core class of circlizePlus, whose objects are generated by the homonymous object constructor ccPlot (). The object of ccPlot acts as a container, holding the objects of ccPar, ccTrack, and ccLink. The objects of ccPar are generated by the homonymous object constructor ccPar(), defining the global parameter for plotting. Users may add one or more ccPar objects to a ccPlot object. circlizePlus aggregates multiple ccPar objects, thus determining the global plotting parameters. A single ccTrack object defines how to plot a track. There is also a subclass of ccTrack named ccGenomicTrack. Their object constructors and functionalities are listed in Supplementary Table S1. A single ccLink object stores data of the link connecting two sectors. There are also 2 subclasses of ccLink: ccHeatmapLink and ccGenomicLink. Their object constructors and functionalities are listed in Supplementary Table S2.

When plotting, users are supposed to call the generic show () function, in which ccPlot objects serve as a parameter. Once the show () function is called, circlizePlus will set global plotting parameters based on ccPar objects stored in ccPlot. Meanwhile, it plots tracks based on stored ccTrack objects or their subclasses, and plots lines based on ccLink objects or their subclasses.

2.4 Relationships between ccTrack, ccTrackGeom, ccCell and ccCellGeom

ccTrackGeom and ccCellGeom stored in ccTrack determine the geometrics plotted in the current track. Data stored in ccTrackGeom works on the entire track, while data stored in ccCellGeom works only on a single cell within the track. ccCellGeom objects cannot be add directly into ccTrack. It must first be added to a ccCell object before adding the ccCell object to ccTrack thereafter. ccCellGeom has a subclass named ccGenomicCellGeom. The object constructors and functionalities of ccTrackGeom and ccCellGeom are listed in Supplementary Table S3.

3 Results

3.1 Workflow of circlizePlus

There are approximately 6 steps for users using circlizePlus to plot (Figure 2), with no strict order restriction for steps 1 to 4.

Step 1: When initiating plot programming, users must first call the ccPlot () function to generate a ccPlot object. Users also need to set a mandatory parameter named "initMode," which can take one of the following 4 values: "initialize," "heatmap.initialize," "initializeWithIdeogram" or "genomicInitialize."

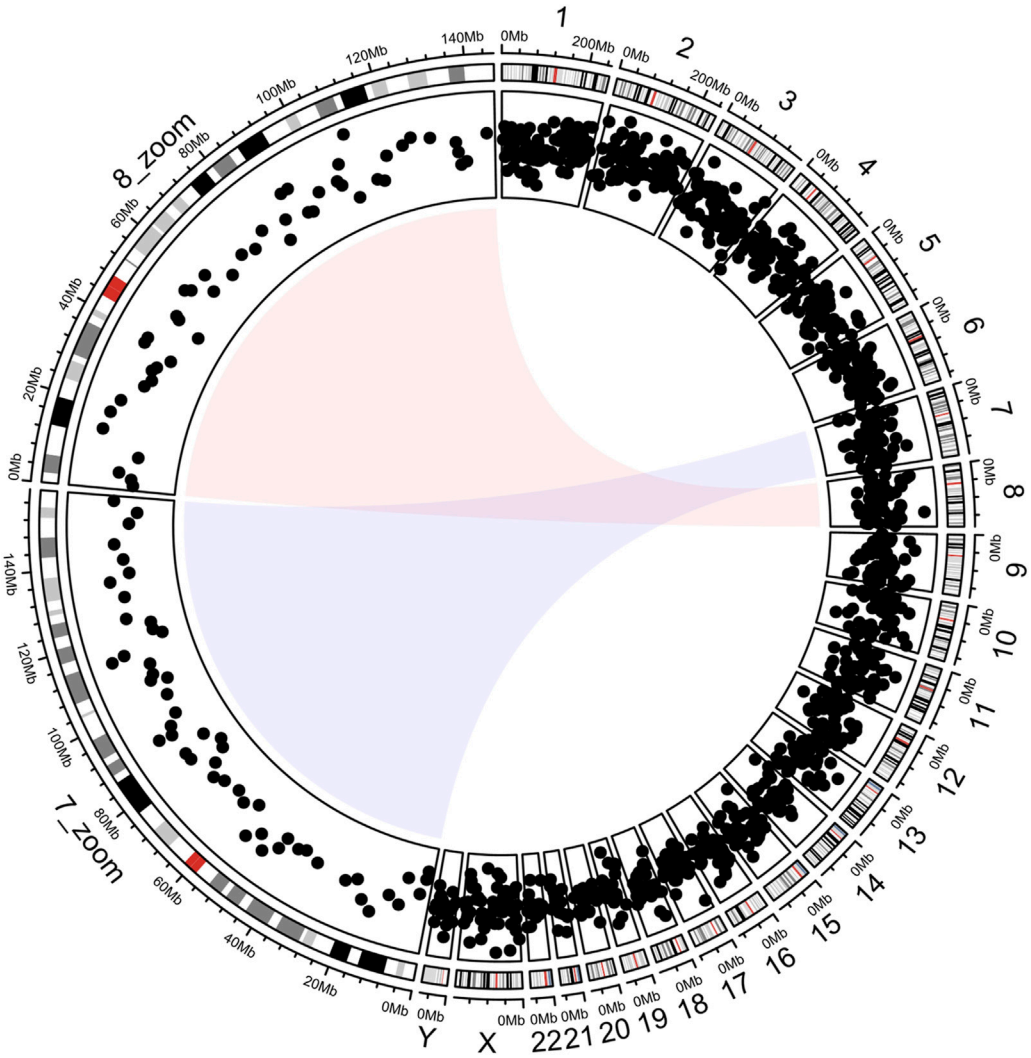

FIGURE 3
Chromosomes 7 and 8 are shown enlarged on the left half of the circle.

Different values of initMode correspond to different plotting scenes. The backend of circlizePlus will call the corresponding initialization functions from the circlize package based on the value of initMode. Therefore, ccPlot () function will have different applicable parameters.

Step 2: For general plotting, users are supposed to generate a ccTrack object. As for visualizing genomic data, users are supposed to create a ccGenomicTrack object.

Step 3: Users are supposed to select the appropriate object constructor function to generate geometric objects, according to the geometrics to be generated.

Step 4: Users are supposed to generate ccLink objects or one of its subclasses, in case users need to connect sectors to represent the relationships between each of them.

Step 5: Users assemble objects according to the addition rules of circlizePlus, and finally call the generic show () function, in which the ccPlot object is used as a parameter to generate a graph. Users are supposed to repeat some steps in steps 1 to 4 and reassemble if the figure generated does not meet expectations.

Step 6: circlizePlus is compatible with the circlize API. If the figure obtained by the above steps is not satisfactory, the user can also call the circlize API for further fine-tuning.

3.2 Example 1: plotted dot plot categorized by chromosome, and zoomed for partial chromosome categorization

This section demonstrates the workflow of circlizePlus based on a small case (Gu et al., 2014). The code lines are segmented according to the workflow defined in the article (codes are shown in Supplementary File S1). It plots the chromosome bands in the outer circle and the corresponding scatter points in the inner circle (Figure 3). Magnified views of chromosomes 7 and 8 are shown on the left parts of the circles.

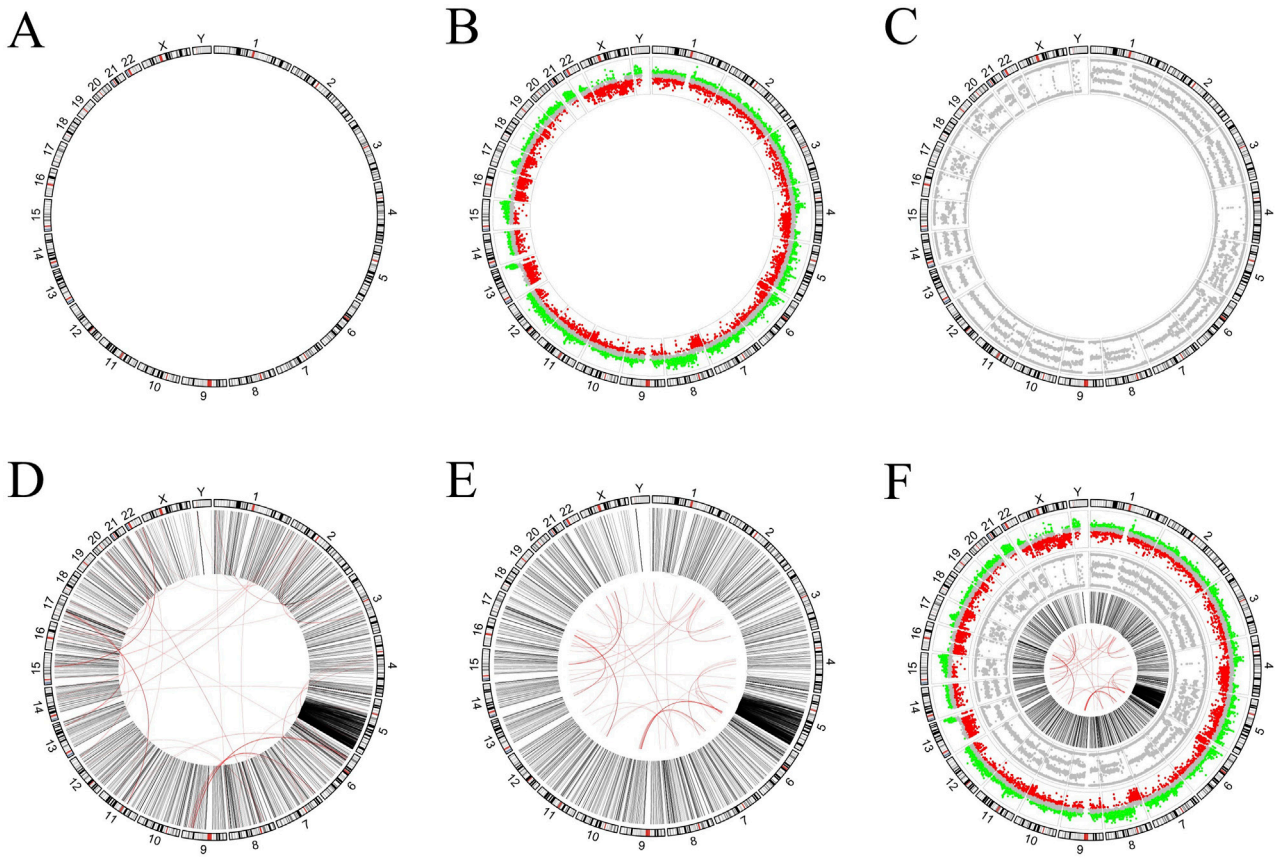

FIGURE 4
Figure generated in example 2. **(A)** Chromosome ideograms in humans. **(B)** The outer ring is the chromosome ideograms, and the inner ring is the dot plot, which is used to represent the number of copies of the gene on the corresponding chromosome. **(C)** The inner ring is also a dot plot that is used to represent the B-allele frequency of the corresponding chromosome. **(D)** The inner loop is a linked line diagram that is used to represent structural variations. **(E)** For aesthetic reasons, the red link lines in Figure D have been scaled. **(F)** The resulting diagram obtained by combining all of the above plots.

The entire code process is like solving an addition problem, and the data flow of drawing can be clearly understood. Note that the data mapping technique is used in step 3. The coordinate parameter of the function ccGenomicPoints () is missing. It will take the coordinate parameter value of the corresponding sector from the variable track1. This technique ensures consistency of coordinate data while reducing code duplication.

4 Discussion

4.1 Why use object-oriented programming

circlizePlus is implemented using object-oriented programming. It is class- and object-centric, with an emphasis on data encapsulation and behavioral reuse. The resulting figure is ultimately determined by the property values of all objects used for drawing, making it easy to maintain and expand.

circlize, on the other hand, is implemented using traditional process-oriented programming, which has functions and processes at its core, and figures are the result of sequential execution of functions. It's suitable for running step by step, with each step calling a function and the corresponding changes will be made immediately on the figure. However, when the parameters that need to be adjusted are used by functions that have been called in some previous steps, the user often has to rerun all the previous code. For example, Figure 3 can be plotted using either circlize or circirclizePlus. When it comes to adjusting the width of the outermost track, circlizePlus needs to run less code, because circlize needs to rerun the code from scratch. In summary, circlizePlus offers advantages in terms of maintenance and scalability, while circlize is better suited for scenarios that require real-time feedback and single-step commissioning (codes are shown in Supplementary File S2).

4.2 Example 2: comparison of two pieces of code that use circlizePlus and circlize to implement the same requirements, respectively

In this section, we try to draw pictures from real publications. The literature (Alves et al., 2013) uses a plot similar to Figure 4 to present the VCaP cancer cell line. The outermost ring corresponds to human chromosome ideograms (Figure 4A). It serves as an X-axis to indicate the position of the data of other rings on the chromosome. The inner ring in Figure 4B represents the number

TABLE 1 Statistics on the amount of code in Example2.

	Figure 4A	Figure 4B	Figure 4C	Figure 4D	Figure 4E	Figure 4F	Total
The number of lines of code implemented with circlize	3	10	6	6	6	16	48
The number of lines of code implemented with circlizePlus	2	8	4	4	4	4	28
The number of characters in the code implemented with circlize (excluding spaces)	118	407	303	277	291	765	2177
The number of characters in the code implemented with circlizePlus (excluding spaces)	100	329	291	175	189	230	1359

of gene copies, and the value of each dot is from the Affymetrix SNP arrays. Its Y-axis range is −1 to 1, where dots from 0.15 to 1 are marked in red, dots from −0.15 to 0.15 are marked in grey, and dots from −1 to 0.15 are marked in green. The inner ring in Figure 4C represents the B-allele frequency (ratio), which ranges from 0 to 1 on the Y-axis. The links plot is used to reflect structural variations (SVs) (Figure 4D). The intra- and inter-chromosomal SVs are on the inner rings and depicted in black and red lines, respectively. The red link lines are obscured by the black link lines because there are quite a few intra-chromosomal SVs. The red link lines have been reduced in scale (Figure 4E) for aesthetic reasons. Combining all of the above plots together gives the picture in the literature (Figure 4F).

The process of drawing Figure 4 is to draw the graphs of the various categories first, and then put them together. Quite a few researchers use this process to create circos diagrams. Both circlize and circlizePlus implement such a process (codes are shown in Supplementary File S3). According to code statistics, circlizePlus uses less code than circlize in such a process. Code statistics show that circlizePlus uses less code than circlize in such a process (Table 1).

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/tianzelab/circlizePlus and https://tianzelab.github.io/circlizePlusBook/.

Author contributions

ZZ: Software, Funding acquisition, original draft.TC: Conceptualization, Software, Writing–original draft. YH: Supervision, Writing–review and editing. YX: Funding acquisition, Writing–review and editing.

Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This work was supported by the key project of Zhejiang Provincial Natural Science Foundation under grant number LZ23A010002 and the National Training Program of Innovation and Entrepreneurship for Undergraduates of Hangzhou Normal University under grant number 202410346057.

Acknowledgments

We are grateful to Dr. Zuguang Gu from DFKZ for his invaluable guidance and support throughout the project.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2025.1535368/full#supplementary-material

References

Alves, I. T., Hiltemann, S., Hartjes, T., van der Spek, P., Stubbs, A., Trapman, J., et al. (2013). Gene fusions by chromothripsis of chromosome 5q in the VCaP prostate cancer cell line. *Hum. Genet.* 132 (6), 709–713. doi:10.1007/s00439-013-1308-1

An, J., Lai, J., Sajjanhar, A., Batra, J., Wang, C., and Nelson, C. C. (2015). J-Circos: an interactive Circos plotter. *Bioinformatics* 31 (9), 1463–1465. doi:10.1093/bioinformatics/btu842

Cheong, W.-H., Tan, Y.-C., Yap, S.-J., and Ng, K.-P. (2015). ClicO FS: an interactive web-based service of Circos. *Bioinformatics* 31 (22), 3685–3687. doi:10.1093/bioinformatics/btv433

Cui, Y., Chen, X., Luo, H., Fan, Z., Luo, J., He, S., et al. (2016). BioCircos.js: an interactive Circos JavaScript library for biological data visualization on web applications. *Bioinformatics* 32 (11), 1740–1742. doi:10.1093/bioinformatics/btw041

Cui, Y., Cui, Z., Xu, J., Hao, D., Shi, J., Wang, D., et al. (2020). NG-Circos: next-generation Circos for data visualization and interpretation. *Nar Genomics Bioinforma.* 2 (3), lqaa069. doi:10.1093/nargab/lqaa069

Cui, Z., Cui, Y., Zang, T., and Wang, Y. (2021). interacCircos: an R package based on JavaScript libraries for the generation of interactive circos plots. *Bioinformatics* 37 (20), 3642–3644. doi:10.1093/bioinformatics/btab232

Darzentas, N. (2010). Circoletto: visualizing sequence similarity with Circos. *Bioinformatics* 26 (20), 2620–2621. doi:10.1093/bioinformatics/btq484

Diaz-Garcia, L., Covarrubias-Pazaran, G., Schlautman, B., and Zalapa, J. (2017). SOFIA: an R package for enhancing genetic visualization with circos. *J. Hered.* 108 (4), 443–448. doi:10.1093/jhered/esx023

Drori, E., Levy, D., Smirin-Yosef, P., Rahimi, O., and Salmon-Divon, M. (2017). CircosVCF: circos visualization of whole-genome sequence variations stored in VCF files. *Bioinformatics* 33 (9), 1392–1393. doi:10.1093/bioinformatics/btw834

Ennis, S., Broin, P. O., and Szegezdi, E. (2023). CCPlotR: an R package for the visualization of cell-cell interactions. *Bioinforma. Adv.* 3 (1), vbad130. doi:10.1093/bioadv/vbad130

Gu, Z., Gu, L., Eils, R., Schlesner, M., and Brors, B. (2014). Circlize implements and enhances circular visualization in R. *Bioinformatics* 30 (19), 2811–2812. doi:10.1093/bioinformatics/btu393

Krzywinski, M., Schein, J., Birol, I., Connors, J., Gascoyne, R., Horsman, D., et al. (2009). Circos: an information aesthetic for comparative genomics. *Genome Res.* 19 (9), 1639–1645. doi:10.1101/gr.092759.109

Marx, H., and Coon, J. J. (2019). MS-Helios: a Circos wrapper to visualize multi-omic datasets. *Bmc Bioinforma.* 20, 21. doi:10.1186/s12859-018-2564-9

Rasche, H., and Hiltemann, S. (2020). Galactic circos: user-friendly circos plots within the galaxy platform. *Gigascience* 9 (6), giaa065. doi:10.1093/gigascience/giaa065

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. New York: Springer-Verlag.

Yu, Y., Ouyang, Y., and Yao, W. (2018). shinyCircos: an R/Shiny application for interactive creation of Circos plot. *Bioinformatics* 34 (7), 1229–1231. doi:10.1093/bioinformatics/btx763

Zhang, H., Meltzer, P., and Davis, S. (2013). RCircos: an R package for Circos 2D track plots. *Bmc Bioinforma.* 14, 244. doi:10.1186/1471-2105-14-244