ggCyto: Next Generation Open-Source Visualization Software for Cytometry

Phu Van, Wenxin Jiang, Raphael Gottardo and Greg Finak

1Vaccine and Infectious Disease Division, Fred Hutchinson Cancer Research Center, Seattle, 98109, USA
2Equal contribution, 3Joint senior authorship

Abstract

Motivation: Open source software for computational cytometry has gained in popularity over the past few years. Efforts such as FlowCAP, the Lyoplate and Euroflow projects have highlighted the importance of efforts to standardize both experimental and computational aspects of cytometry data analysis. The R/BioConductor platform hosts the largest collection of open source cytometry software covering all aspects of data analysis and providing infrastructure to represent and analyze cytometry data with all relevant experimental, gating, and cell population annotations enabling fully reproducible data analysis. Data visualization frameworks to support this infrastructure have lagged behind.

Results: ggCyto is a new open-source BioConductor software package for cytometry data visualization built on ggplot2 that enables ggplot-like functionality with the core BioConductor flow cytometry data structures. Amongst its features are the ability to transform data and axes on-the-fly using cytometry-specific transformations, plot faceting by experimental meta-data variables, and partial matching of channel, marker and cell populations names to the contents of the BioConductor cytometry data structures. We demonstrate the salient features of the package using publicly available cytometry data with complete reproducible examples in a supplementary material vignette.


Contact: gfinak@fredhutch.org

Supplementary information: Supplementary data are available at Bioinformatics online and at http://rglab.org/ggcyto/.

Introduction

Cytometry (FCM) is the primary assay for immune monitoring in clinical and research applications (Maecker et al., 2012). Pipelines must handle preprocessing, quality control, analysis (i.e., cell clustering or manual partitioning into homogeneous groups) (O’Neill et al., 2013; Saey et al., 2016) and visualization. Proprietary platforms, including FlowJo (Ashland, OR), WinList, FCSExpress, and DIVA are the de-facto standards for end-to-end FCM data analysis. Other programming frameworks like Matlab (Mathworks) and Mathematica (Mathematica. 9.0. Champaign, IL: Wolfram Research) provide functionality for data import and exploration (indeed, SPADE (Qiu et al., 2011) was initially developed for MATLAB), but lack the general abstraction of cytometry-specific data structures helpful for data analysis. Open-source projects like R/BioConductor (R/BioC) (Gentleman et al., 2004; Ihaka and Gentleman, 1996) and Python provide FCM functionality through user-contributed packages (Frelinger et al., 2012). Currently 47 open source software packages in BioConductor are tagged for “FlowCytometry” (http://bioconductor.org/packages/release/BiocViews.html) but only flowViz (Sarkar et al., 2008) is visualization-centric and doesn’t support the core BioConductor data structures used to store analyzed, gated and annotated, single-cell FCM data (see Supplementary Information (S.I.)). Other packages focus on different aspects of automated analysis.

We introduce ggcyto, a BioConductor package for building reproducible FCM visualizations programmatically. It is built on ggplot2 (Wickham, 2009) and supports the core BioConductor cytometry data structures making it compatible with any package using those structures (see S.I.).
Van et al.

ggcyto is two-fold: flow parameters and data source are decisions left to the user. Leveraging ggcyto’s context-aware, selecting objects (for ungated data) or data source (for gated data, Figure 1 and S.I.). In the case of ungated data, the user specifies the channels / markers to visualize, rather than the cell population (since the latter is not defined).

Customizing plots with cytometry-specific layers.

The ggcyto() API provides greater flexibility and customization than autoplot (Figure 1). When using ggcyto, the layers and defaults selected by autoplot are decisions left to the user. Leveraging ggcyto’s cytometry-specific layers and geoms, the user builds the plot (Figure 1 and S.I.) to include the gates, overlays (e.g. backgating), data or axis transformations, cell subpopulations, and cell subpopulation statistics of interest, and specifies the faceting of plots by metadata annotations (S.I.). The ggcyto API can be particularly useful to project cell populations onto other markers (i.e., not necessarily those on which the populations are defined).

The support for data transformations in ggcyto is two-fold: ggcyto can transform the underlying data (Figure 1), or it can transform the axes using the transformation stored in the data source (Figure 1). These approaches are demonstrated in the S.I.

Examples

The functionality of ggcyto is demonstrated using the Lyoplate data set from FlowCAP 4 (Finak et al., 2016) available in the flowWorkspaceData R/BioConductor package and on the ImmuneSpace portal (Brusic et al., 2014) (see the S.I. for link to this data on ImmuneSpace), as well as the graft vs. host disease (GvHD) data available in the flowCore R/BioConductor package. Reproducible examples with R code are available in the S.I. and available at http://rlab.org/ggcyto. In future additional cytometry data may be available via the more modern AnnotationHub or ExperimentHub resources (Morgan et al., 2016; Pasolli et al., 2017).

Conclusion

The ggcyto package provides a powerful and unified visualization interface to complex, ungated or gated, annotated cytometry data structures and provides a key component of a reproducible research workflow. Specifically, the package allows for easy visualization of specific cytometry cell populations and gates, on the fly data and axis transformation, back-gating visualization, and easy faceting by study metadata in order to explore variability in an experiment. User-friendliness is made possible through fuzzy name matching, lazy data loading, and context-sensitive behavior that aims to capture “what the user means to do” most frequently. Areas for future developments are highlighted in the S.I.

Acknowledgements

The authors wish to acknowledge the contributions of the computational flow community for testing and feedback on this software package.

Funding

This work was supported by an NIGMS grant, [R01 GM118417-01A1] (to GF), and a grant from the Bill and Melinda Gates Foundation, [OPP1032317] (to RG). Conflict of Interest: none declared.

References


