cellxgene VIP unleashes full power of interactive visualization, plotting and analysis of scRNA-seq data in the scale of millions of cells

Authors: Kejie Li[1‡], Zhengyu Ouyang[2‡], Dongdong Lin[1], Michael Mingueneau[1], Will Chen[1], David Sexton[1], Baohong Zhang[1*]

Affiliations:

[1]Research Department, Biogen, Inc., 225 Binney St, Cambridge, MA 02142, USA

[2]BioInfoRx, Inc., 510 Charmany Dr, Suite 275A, Madison, WI 53719, USA

Emails:

KL: kejie.li@biogen.com

ZO: oyoung@bioinforx.com

DL: dongdong.lin@biogen.com

MM: michael.mingueneau@biogen.com

WC: wwchen@post.harvard.edu

DS: david.sexton@biogen.com

BZ: baohong.zhang@biogen.com

‡ These authors contributed equally to the work

* Correspondence to Baohong Zhang, baohong.zhang@biogen.com

Abstract

To meet the growing demands from scientists to effectively extract deep insights from single cell RNA-seq datasets, we developed cellxgene VIP, a frontend interactive visualization plugin to cellxgene framework, which directly interacts with in-memory data to generate a comprehensive set of plots in high resolution, perform advanced analysis, and make data downloadable for further analysis. It makes large scale scRNA-seq data visualization and analysis more accessible and reproducible with the potential to become an ecosystem for the scientific community to contribute even more modules to the Swiss knife of scRNA-seq data exploration tool.

Introduction

Since the first single-cell RNA sequencing (scRNA-seq) study was debuted in 2009[1], over 1080 scRNA-seq studies have been published to date. At least 33 studies have reported profiles in 200k or more cells[2]. The largest scRNA-seq study reported 2.5 million mouse cells[3]. It is foreseen that there will be a trend of increasing cell size of 500k or more in scRNA-seq studies. The sheer amount of data had brought challenges in visualizing and exploring such big data set interactively for scientists, even computational biologists.

Cellxgene[4] is a leading open source scRNA-Seq data visualization tool recommended in a recent evaluation[5], which scales well in millions of cells and scores high in user experience by leveraging modern web techniques with interactive features. Tested by a large community of Biogen biologists, cellxgene works the best in meeting the need of handling data themselves except lack of some essential plotting seen in scRNA-seq related publications and analysis functions that biologists are used to, hinders its utility and limits scientists from taking advantage of ever accumulating scRNA-seq data in the public to its full potential.

Result

To fill the gap, we developed a plugin of cellxgene named Visualization in Plugin, in short VIP, to address the urgent needs of such essential functions for interactive visual exploration and generation of publication-ready figures. Notably, it enhanced plotting functions significantly to generate violin, stacked violin, stacked bar, heatmap, volcano, embedding, dot, track, density, 2D density, sankey and dual-gene plot in high-resolution by calling server-side scanpy's[6] plotting functions and general plotting libraries as illustrated in Figure 1 and Supplementary Tutorial. Innovative split plots in embedding and density plots by annotations such as cell types make it easier to see which genes are more strongly expressed in which cells (Supplementary Fig. S10 and Fig. S13). With the goal of becoming a platform to bridge users and various tools, cellxgene VIP not only leverages python analytical capabilities, but also opens to another popular language R demonstrated by volcano plot (Supplementary Fig. S19). In addition, interactive Sankey diagram and stacked barplot based on yet another language, JavaScript enable investigation of relationship between metadata and gene expression (Supplementary Fig. S16 and Fig S17).

Beyond the visualization functions, we integrated diffxpy[7] to provide analytical capabilities in VIP. In addition to simple t-test and limited number of differentially expressed (DE) genes in the original tool, users can get an unlimited number of DE genes by all of the methods implemented in diffxpy. Volcano plot with fold change and p-value is also provided to enable global view of biological perturbation (Fig. S19). To facilitate reproducible research, sessions can be saved and loaded to preserve previous analysis settings with option to pre-set default embedding and color-by option when the session is loaded (Supplementary for more details).

VIP provides easy-to-use server-side installation script of cellxgene with customized minimal modifications. It took advantage of the cached data by cellxgene server to avoid data reloading that is quite time consuming for large data set in gigabytes. On the client-side, a non-

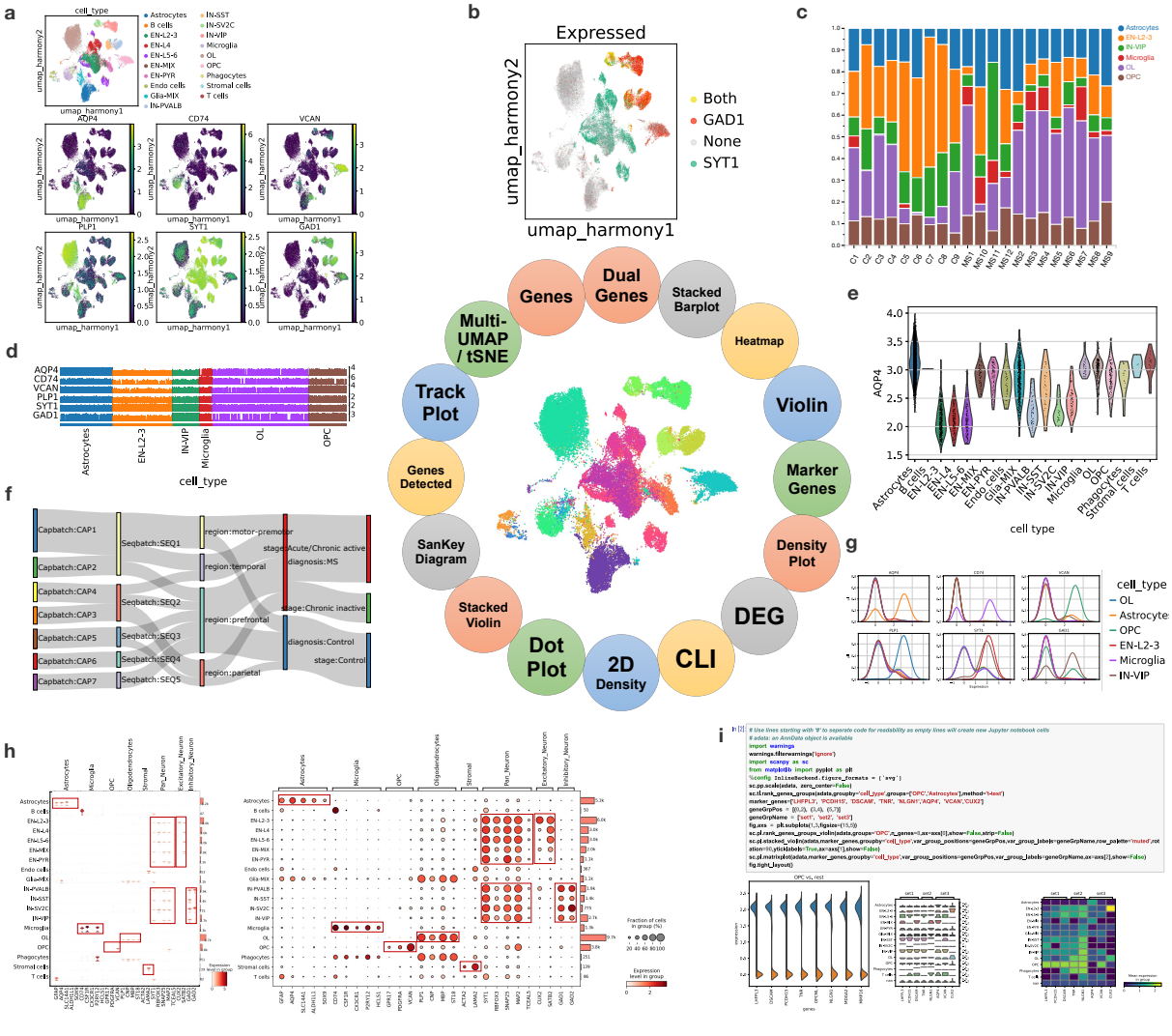


Figure 1 | cellxgene VIP serves as an ecosystem of plugins which provide essential functions for publication-ready, interactive visualization, as well as Command Line Interface (CLI) for analytics. (svg files were assembled by figureComposer[8] with zoomable version available at https://bit.ly/2QqdMg3 that is best viewed by Chrome) (**a**) **Multi-tSNE/UMAP plot** visually highlights which cells expressing cell markers on selected embedding (UMAP based on harmony batch correction in this example). (**b**) **Dual-gene plot** highlights cells express SYT1 and GAD1 (green SYT1 only, red GAD1 only, yellow co-expression of STY1 and GAD1), expression cutoff 2.2. (**c**) **Stacked barplot** demonstrates the fraction of each major cell type across each sample (C are Control and MS are MS patients). (**d**) **Trackplot** shows expression of lineage marker genes across individual cells in annotated clusters. (**e**) **Violin plot** shows the AQP4 gene expression across cell types. (**f**) **Sankey diagram** (a.k.a. Riverplot) provides quick and easy way to explore the inter-dependent relationship of variables in the MS snRNAseq dataset[9]. (**g**) **Density plots** shows expression of marker genes across annotated clusters and split across cell types. (**h**) **Stacked violin** and **Dot plot** are the key visualizations of selected cell markers across cell types. They highlight their selective expression and validates the scRNAseq approach and visualization method. (**i**) **CLI** exposed by mini Jupyter Notebook to provide maximal flexibility of performing various analytics on the whole or sliced single cell dataset.

invasive JavaScript panel was elegantly plugged in to bridge the communications between the user and server through web browser store (Supplementary Methods). With extensibility in mind, we decoupled the development of the plugin and cellxgene as much as possible, so it is easy to incorporate additional tools in Python or any other languages like R on server side while taking full benefits of interactive user experience offered by both cellxgene and the plugin. The VIP inherited all cellxgene functions and goes beyond through real time communication with cellxgene by co-localization on the same web page. Within such design, developers can easily add new functions to the plugin without touching cellxgene code at all.

Loading half-million-cell dataset even with sparse matrix compression applied would take up approximately 20G of memory. Standard Mac/PC laptop would not be able to load such a dataset into memory, not to mention doing additional interactive manipulation and plotting. Within cellxgene VIP, all heavy lifting functions such as differential gene expression analysis are executed on the server end. It greatly reduces the burden on the client browser side.

To further empower advanced users with programming skills, a Command Line Interface (CLI) is provided to offer unlimited analytical potentials by calling any Python and/or R packages such as Seurat[10] (sample vignette in GitHub) installed on the server directly through a mini Jupyter notebook environment (Supplementary Fig. S21).

In addition to scalability, versatility and flexibility, we paid great attention to user-friendliness by crafting the plugin to allow users to change the look-n-feel of VIP, set figure options like font size, image format and resolution, color palette for embedding (Supplementary Fig. S5), initialize cellxgene embedding, coloring and brief description of the data (Supplementary Methods), sync cell selection, gene input from main window to VIP automatically, save and load sessions, and create combinatorial annotations and abbreviations (Supplementary Fig. S22). Further, we made cellxgene VIP open source and free of charge to facilitate broader scientific collaborations on large scale scRNA-Seq data sets.

In summary, our web-based interactive tool built upon cellxgene but greatly extended its

plotting and analytical capabilities by integrating state-of-the-art tools in this field. It allows users with no programming experience to rapidly explore scRNA-seq data and create high-resolution figures commonly seen in high-profile publications. Furthermore, it is the first tool to the author's knowledge to allow computational biologists to write their own code to communicate with the hosting server via a mini Jupyter notebook like interface. It opens up unlimited capabilities even beyond the rich set of plotting functions provided in the tool.

Code availability

The demo of the tool is available at https://cellxgenevip-ms.bxgenomics.com while the source code and detailed installation guide is provided at https://github.com/interactivereport/cellxgene_VIP. Cellxgene VIP is released under the MIT License.

Data availability

The demo data in h5ad format can be reproduced by following the Jupyter notebook https://bit.ly/2CeUHtO or downloaded from https://bit.ly/2F5RUnQ.

Acknowledgements

The authors are grateful and indebted to the cellxgene team from Chan Zuckerberg Initiative for the development of the base framework. We would also like to thank L. Schirmer and D.H. Rowitch for allowing us to use scRNA-seq data on Multiple Sclerosis for the demonstration of cellxgene VIP.

Author contributions

K.L, Z.O. and B.Z. designed and implemented the software. All authors contributed testing of the tool. K.L., D.L., M.M. and B.Z. wrote the manuscript with input from all authors. All authors read and approved the manuscript.

Competing interests

The authors declare no competing interests.

References

1. Tang, F. et al. mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods* **6**, 377-382 (2009).
2. Svensson, V., da Veiga Beltrame, E. & Pachter, L. A curated database reveals trends in single-cell transcriptomics. *bioRxiv*, 742304 (2019).
3. Rodriques, S.G. et al. Slide-seq: A scalable technology for measuring genome-wide expression at high spatial resolution. *Science* **363**, 1463 (2019).
4. Chan Zuckerberg Initiative chanzuckerberg/cellxgene: An interactive explorer for single-cell transcriptomics data. *Available at https://chanzuckerberg.github.io/cellxgene/* ([Accessed: 5 May 2020]).
5. Cakir, B. et al. Comparison of visualization tools for single-cell RNAseq data. *NAR Genomics and Bioinformatics* **2** (2020).
6. Wolf, F.A., Angerer, P. & Theis, F.J. SCANPY: large-scale single-cell gene expression data analysis. *Genome Biology* **19**, 15 (2018).
7. David S. Fischer, F.R.H. Fast and scalable differential expression analysis on single-cell RNA-seq data. *Available at https://github.com/theislab/diffxpy* ([Accessed: 8 May 2020]).
8. Li, K. et al. figureComposer: A web-based interactive multi-panel bio-infographic designing tool. *bioRxiv*, 2020.2003.2004.976589 (2020).
9. Schirmer, L. et al. Neuronal vulnerability and multilineage diversity in multiple sclerosis. *Nature* **573**, 75-82 (2019).
10. Stuart, T. et al. Comprehensive integration of single-cell data. *Cell* **177**, 1888-1902. e1821 (2019).

Supplementary Materials of cellxgene VIP

Web URLs of tools and online user guide

cellxgene VIP source code: https://github.com/interactivereport/cellxgene_VIP

cellxgene VIP demo sites: https://cellxgenevip-ms.bxgenomics.com , Schirmer / Rowitch MS, Human brain snRNAseq 46k cells, Nature 2019 Schirmer et al.

cellxgene: https://github.com/chanzuckerberg/cellxgene

cellxgene tutorial: https://cellgeni.readthedocs.io/en/latest/visualisations.html

cellxgene features: https://chanzuckerberg.github.io/cellxgene/posts/gallery

cellxgene data preparation: https://chanzuckerberg.github.io/cellxgene/posts/prepare.html

scanpy: https://github.com/theislab/scanpy

scanpy plots: https://scanpy-tutorials.readthedocs.io/en/latest/visualizing-marker-genes.html

diffxpy: https://github.com/theislab/diffxpy

diffxpy tests: https://diffxpy.readthedocs.io/en/latest/tutorials.html#differential-testing

Benchmarking of interactive data visualization of single-cell RNAseq data — Batuhan Cakir : https://www.youtube.com/watch?v=3nH2xi_Ni6I

sceasy: convertor of scRNA-Seq data formats https://github.com/cellgeni/sceasy

jupytext: https://jupytext.readthedocs.io , Jupyter Notebooks as Markdown Documents, Julia, Python or R Scripts

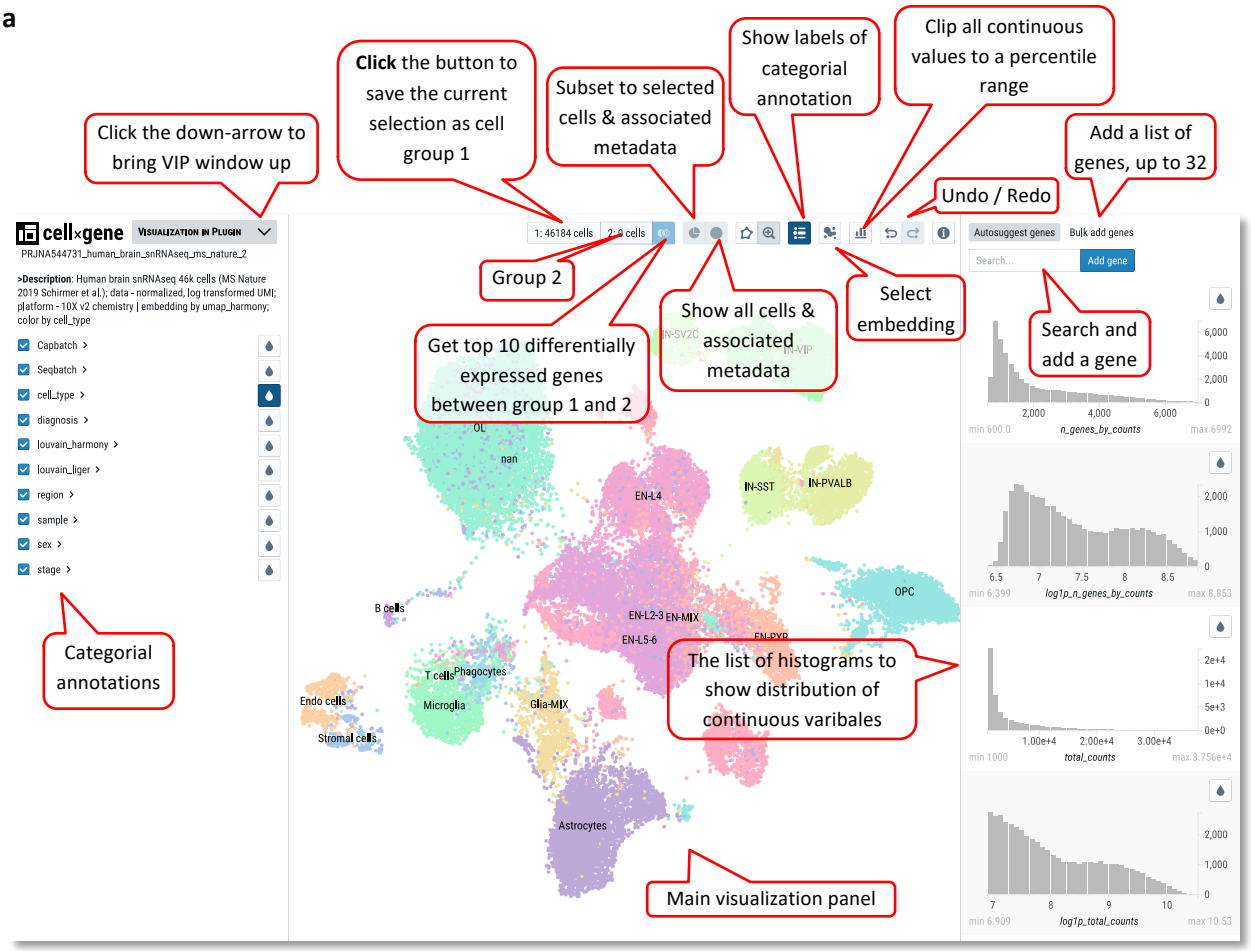
nbconvert: https://nbconvert.readthedocs.io , Convert a Jupyter .ipynb notebook document file into another static format including HTML, LaTeX, PDF, Markdown, and more

Sup. Tutorial – How to use cellxgene VIP

1. Graphical user interface of cellxgene and VIP

The main window of cellxgene is divided into three regions, the left panel mainly displays categorial annotations, brief description of the data set and initial graphics setting, specifically embedding and coloring of cells. On the right panel, it hosts continuous variables, such as qc metrics shown in histogram with x, y corresponding to values of a measurement and numbers of cells, respectively. More importantly, cells shown as individual dots are presented in the center panel based on a selected embedding and colored by either categorial annotations or continuous variables, which is indicated by pressed rain drop icon, e.g., cell_type in Fig. S1a.

a



b

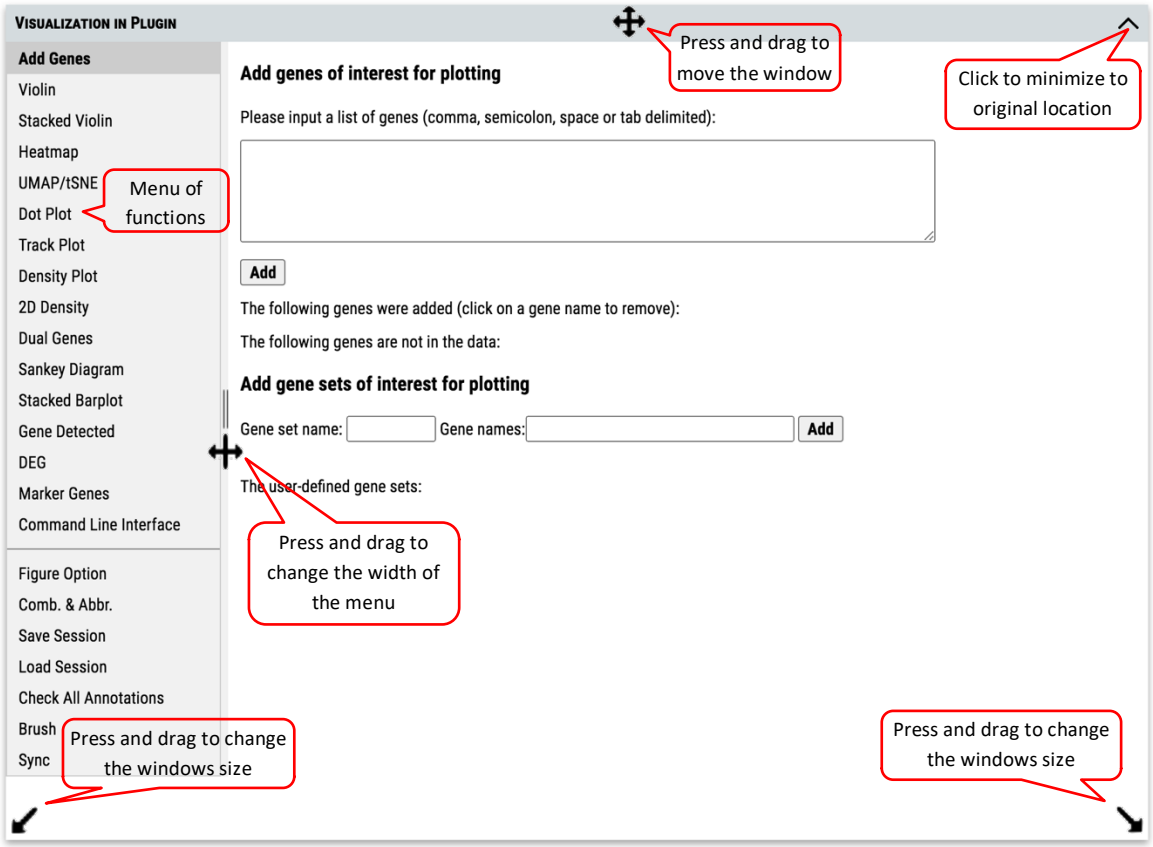


Fig. S1 | (a) Cellxgene main window, functional icons and minimized VIP bar next to cellxgene logo. (b) VIP (Visualization in Plugin) window and controls of user interface. The cursor will change to corresponding icon when mouse hovers over control anchors inside the window. In the case of missing title bar after operation, changing the size of outside browser window (not VIP window) will always bring the VIP window back to the original location near the cellxgene logo.

2. Cell selection by categorial annotations

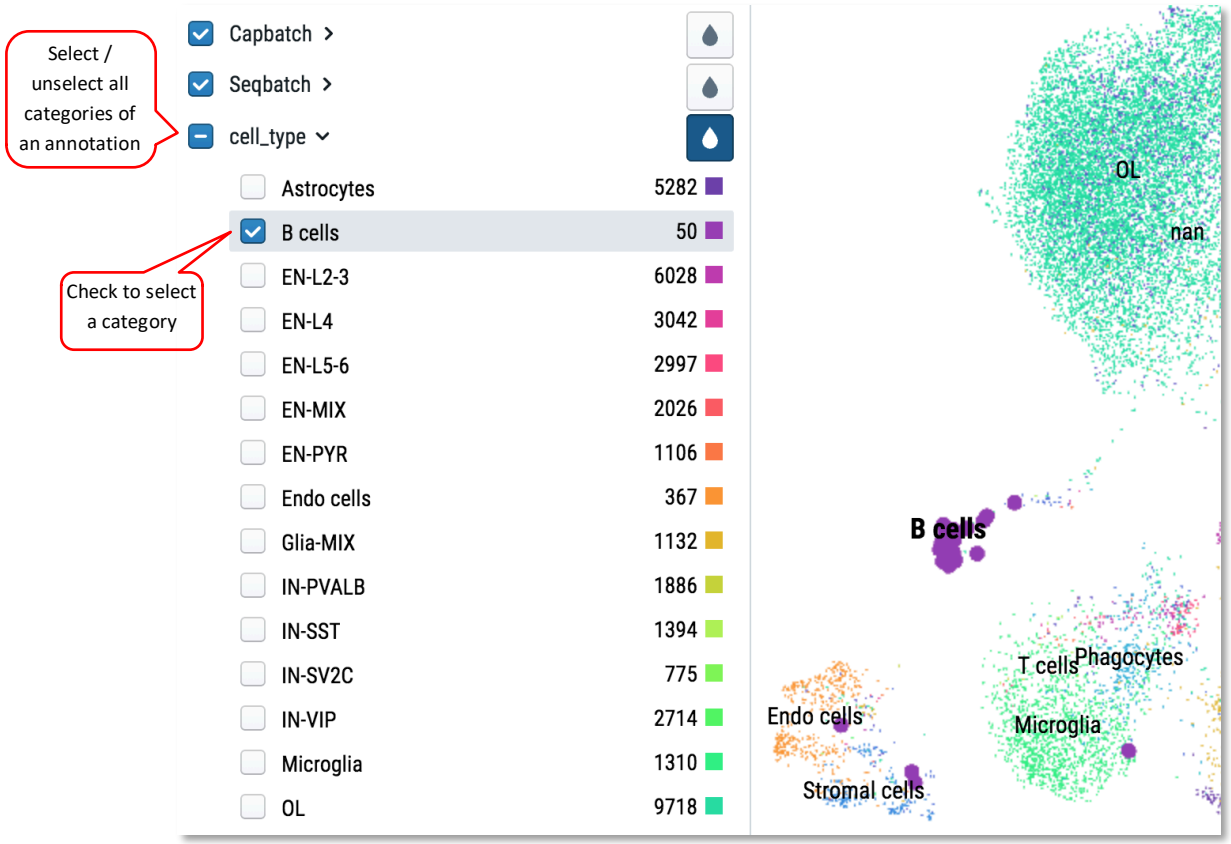


Fig. S2 | Cell selection by categorial annotation. Selected B cells are shown in bold dots and highlighted in purple color when hovering mouse over the cluster.

It is an overlap operation when categories from multiple annotations are checked to make the final selection. E.g., if male from sex is also checked besides B cells, it means cells from B cells cluster of male samples are selected.

Note: Click “1:” or “2:” button to save cell selection into group 1 or 2.

3. Cell selection by brushing on distribution of continues variables

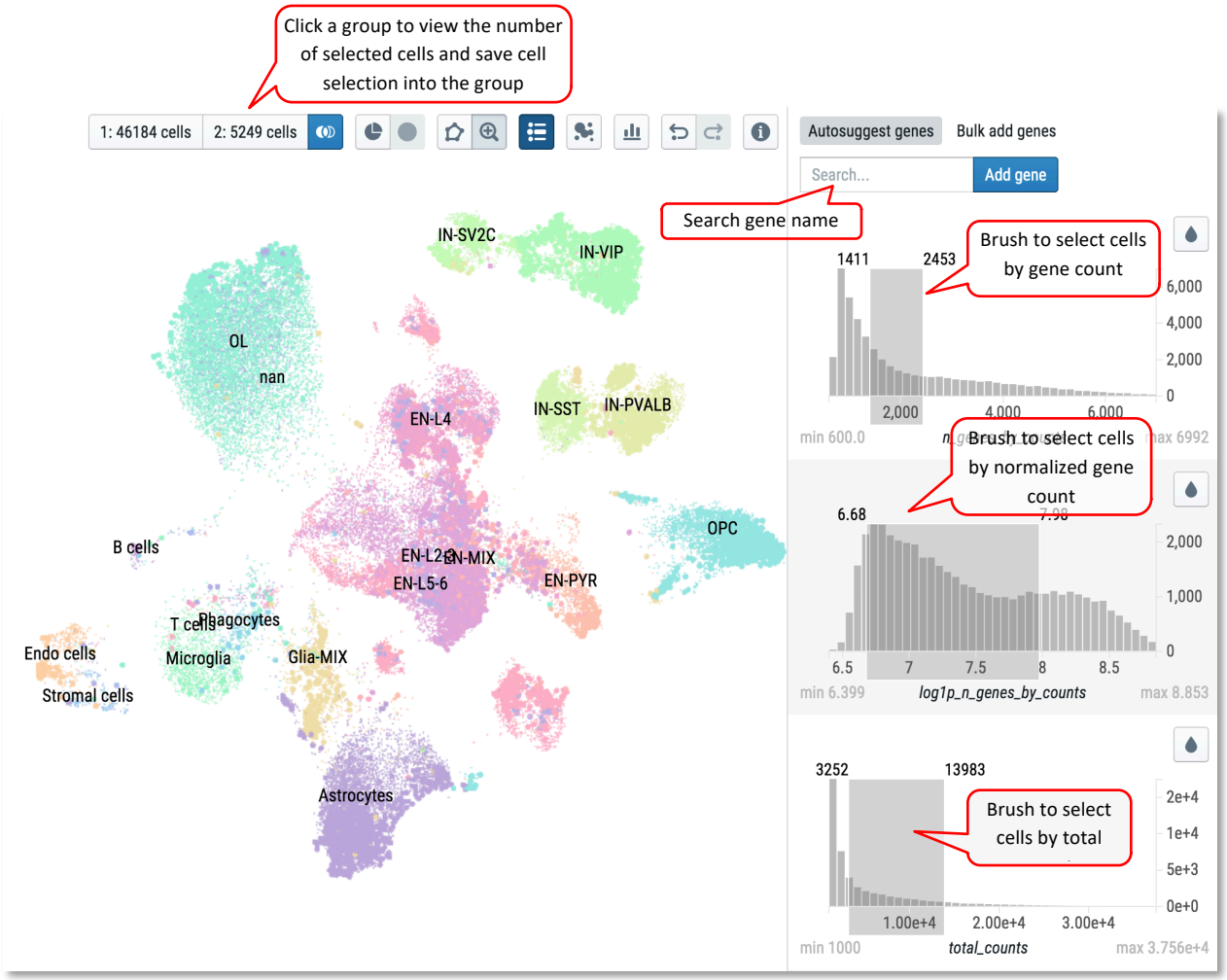


Fig. S3 | Cell selection by brushing the ranges of continuous variables. Low- and high-end values are shown at top corners of brushing boxes in dark gray.

Note: Histograms of expression values of genes can by brushed as well to get cells expressing certain genes in the range.

4. Free hand Lasso selection on dots representing cells

From the cell visualization panel, user can freely select a cluster of cells of interest by using 'Lasso' selection tool. The selected cluster of cells can also be added as a group for downstream analysis in cellxgene VIP.

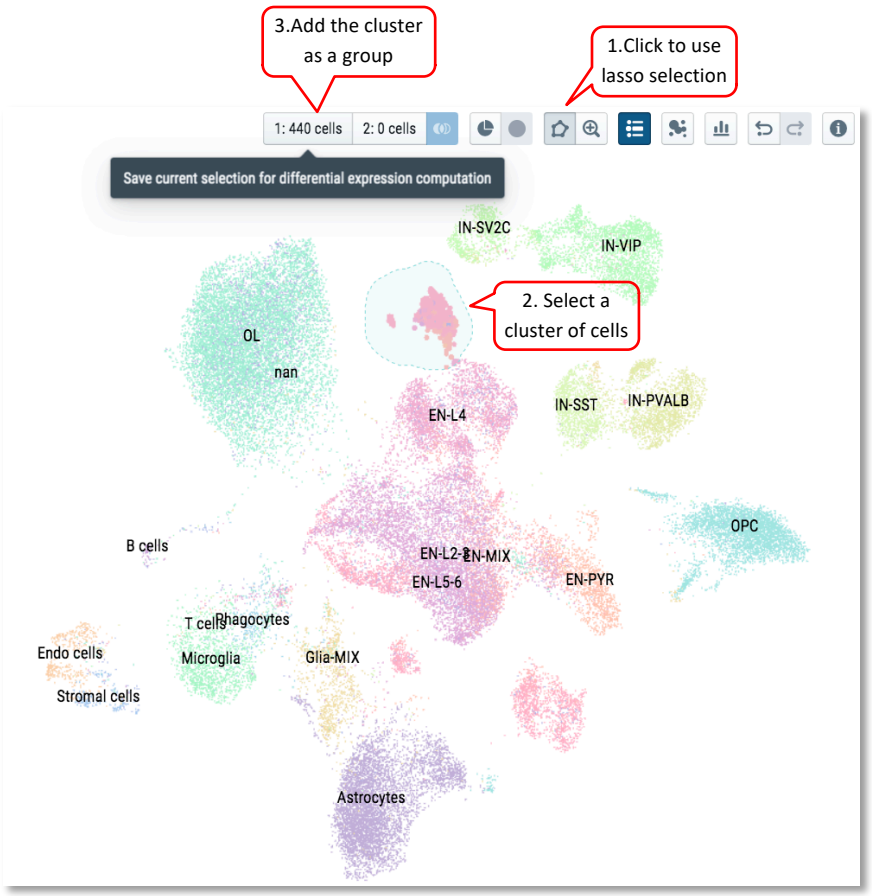


Fig. S4 | Select cells by using free hand Lasso selection tool and add these cells as a group for further analysis in cellxgene VIP.

Note: Please try to draw as close as possible to the starting point in the end to make an enclosed shape to ensure successfully lasso selection.

5. VIP – Figure Option

User can set parameters for figure plotting that control plotting functions except CLI. 'split_show' branch of Scanpy offers better representation of Stacked Violin and Dot Plot comparing to master branch.

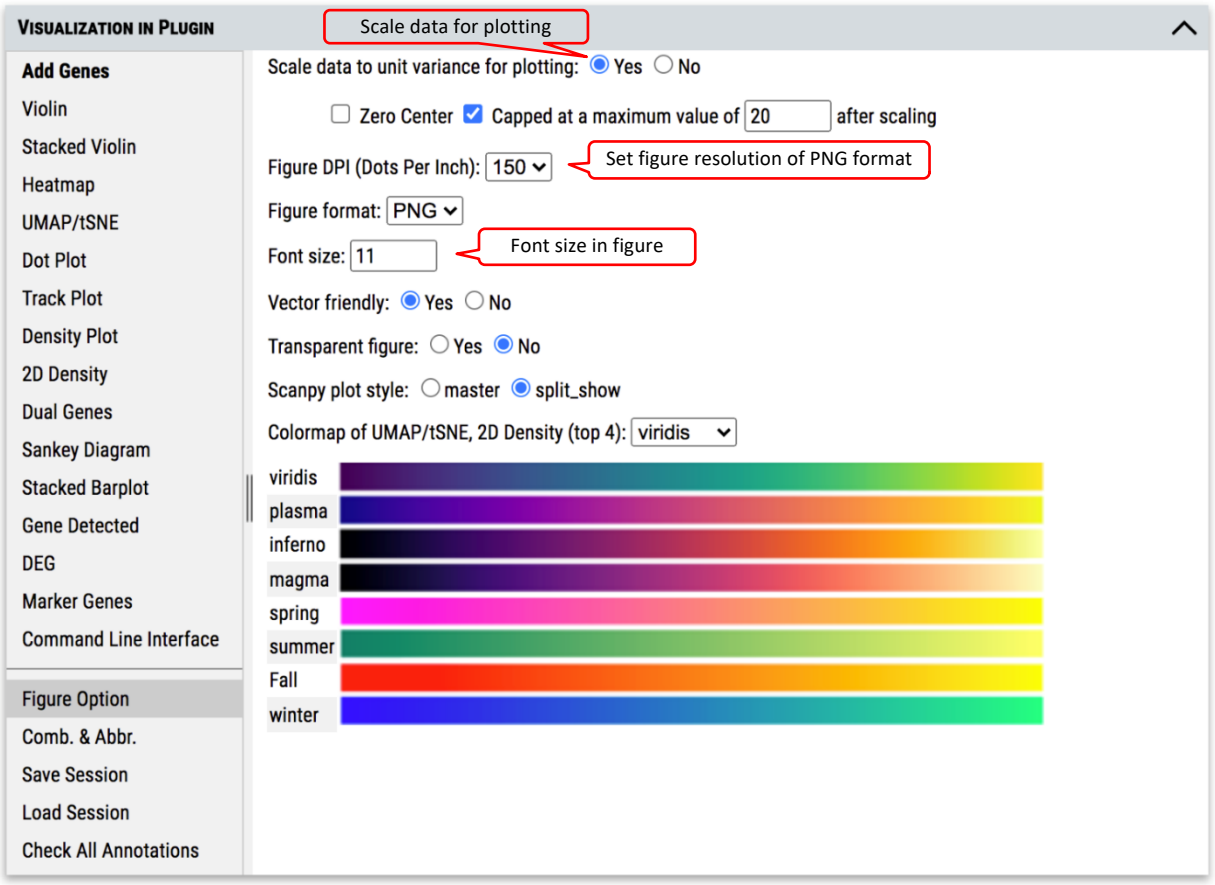


Fig. S5 | Setting parameters for figure plotting.

Scaled data have zero mean and unit variance per gene. This was performed by calculating z-scores of the expression data using Scanpy's scale function. (Scanpy pp.scale function: Scale data to unit variance and zero mean.)

We provide flexibility to allow 1) scale to unit variance or not; 2) Zero centered or not; 3) Capped at max value after scaling.

We recommend using scaled data for plotting/visualization while using non-scaled data for differential gene expression analysis.

Note: Dot plot is one exception in visualization category which uses non-scaled data for meaningful interpretation.

6. VIP – Add Genes / Gene Sets

Cellxgene VIP allows user to add any genes or gene sets for extensive exploration and visualization. User can either type a list of gene in the textbox or create sets of genes to be grouped together in plots. Then the genes will be automatically listed for plotting in other functional modules after checking availability in the dataset.

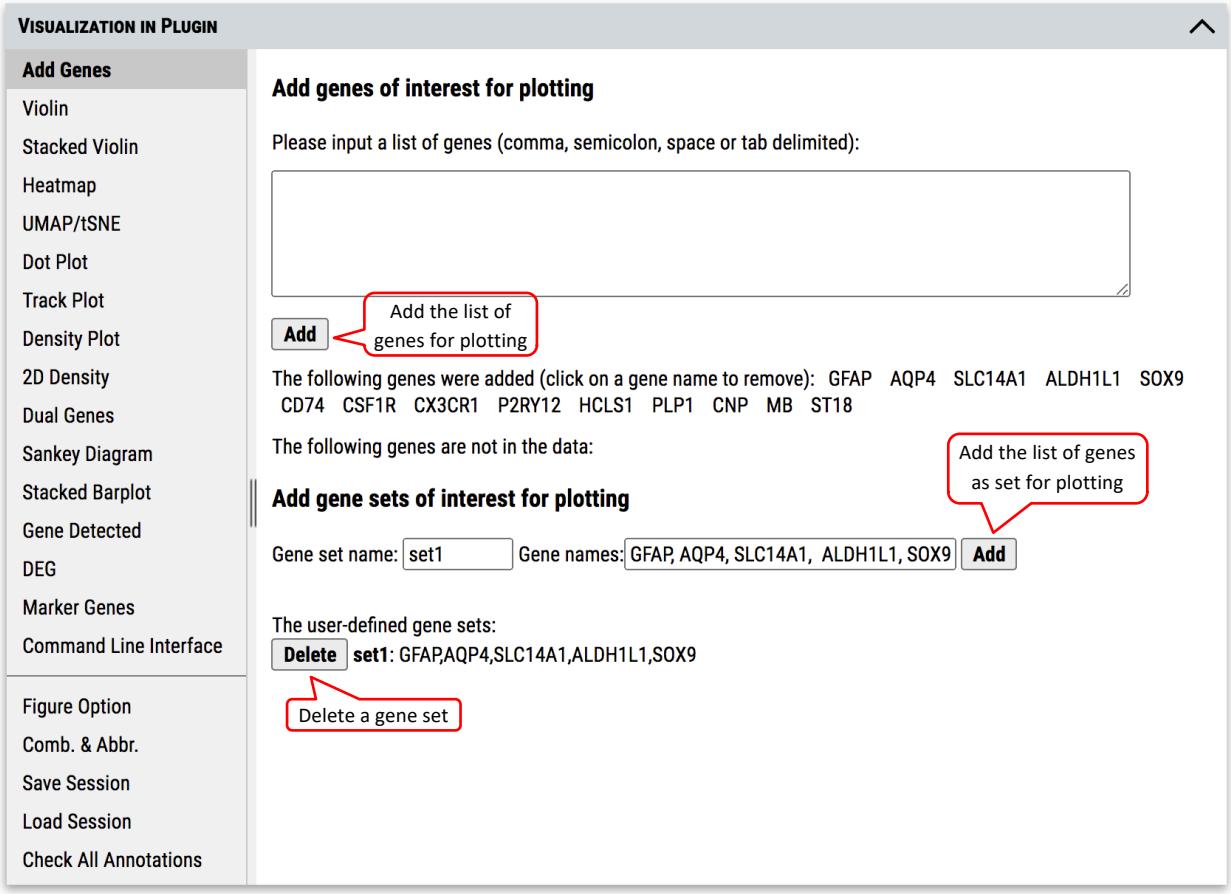


Fig. S6 | Add gene or gene sets for plotting.

Note: The cursor will turn to cross icon while hovering over a gene name, then click to delete the gene.

7. VIP – Violin Plot

To plot expression of gene among categories of an annotation, e.g., cell type, sex, or batch etc.

Step 1. User needs to select the group(s) of cells for plotting. These groups can be created by using selection tools illustrated in tutorial section 2, 3 and/or 4. Initially, all of cells are gathered in 'Group 1' by default.

Step 2. Select a gene from the gene list which could be added as shown in section 6. An expression level cutoff can be set to further filter out cells with low level expression of such gene.

Step 3. Select the annotation to group cells for plotting.

Step 4. Execute plotting, get plotting data (i.e., gene expression), manipulate image (e.g., zoom in/out) or save the image.

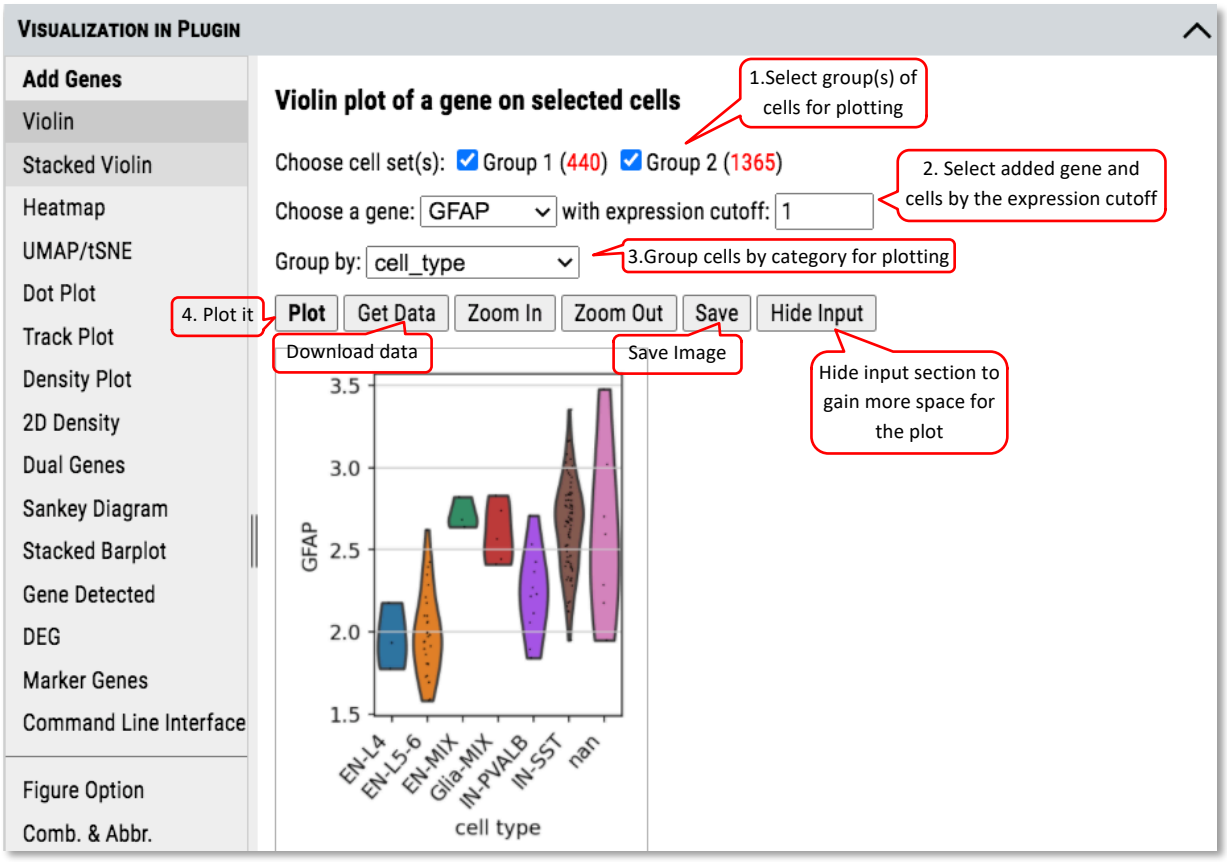


Fig. S7 | Violin plot of gene expression values of a gene grouped by cell type.

Note: Figure resolution and format can be set in "Figure Option" tab as shown in tutorial section 5.

8. VIP – Stacked Violin

Beyond plotting expression values of a gene, stacked violin allows plotting of multiple genes together.

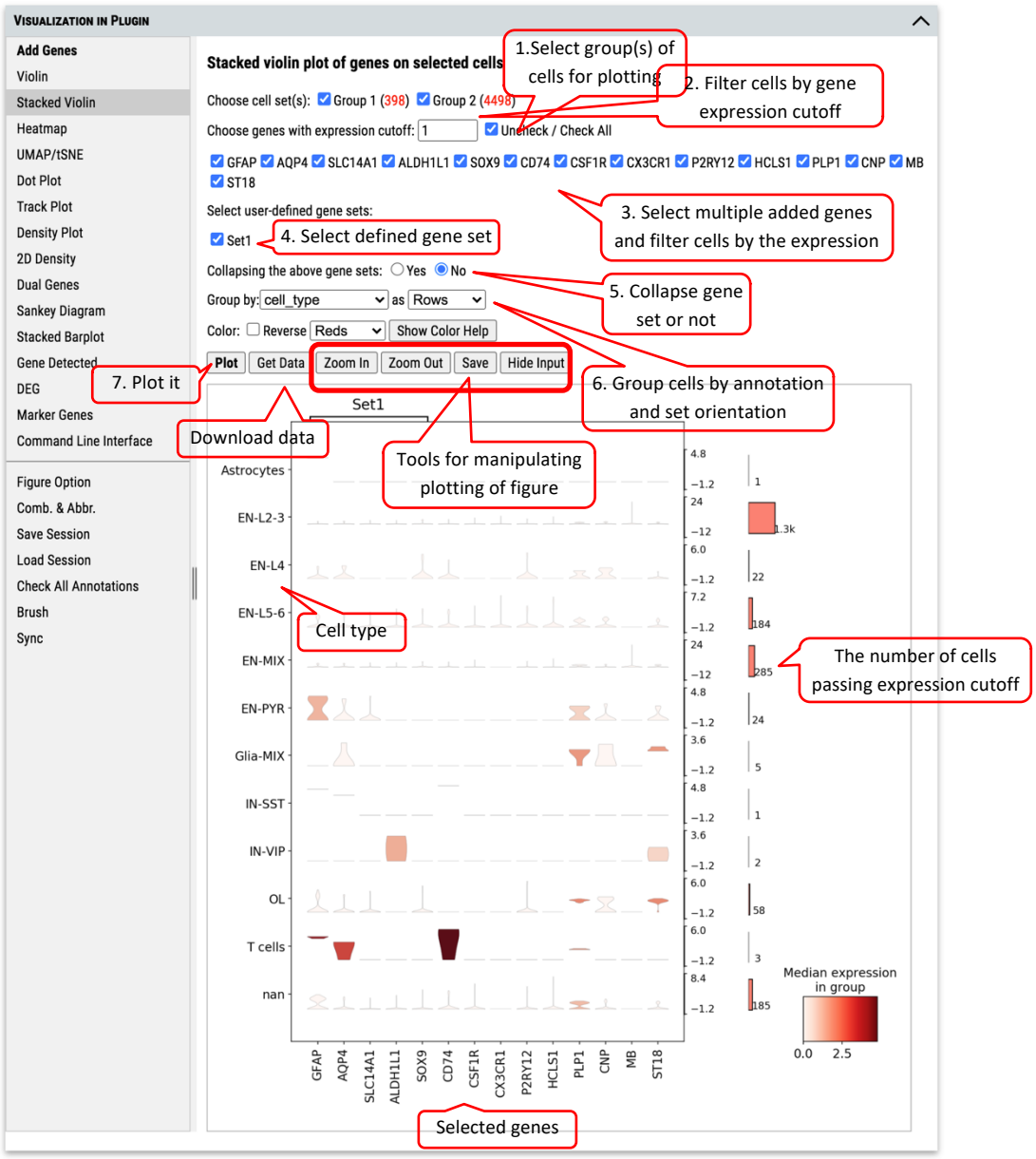


Fig. S8 | Stacked Violin plot of multiple genes and/or gene set.

Note: If collapsing of gene sets is set to 'Yes', average gene expression of genes in a set is used for plotting.

9. VIP – Heatmap

To show or compare the expression level (i.e., expression value or expression Z-score) of multiple genes among the selected group of cells.

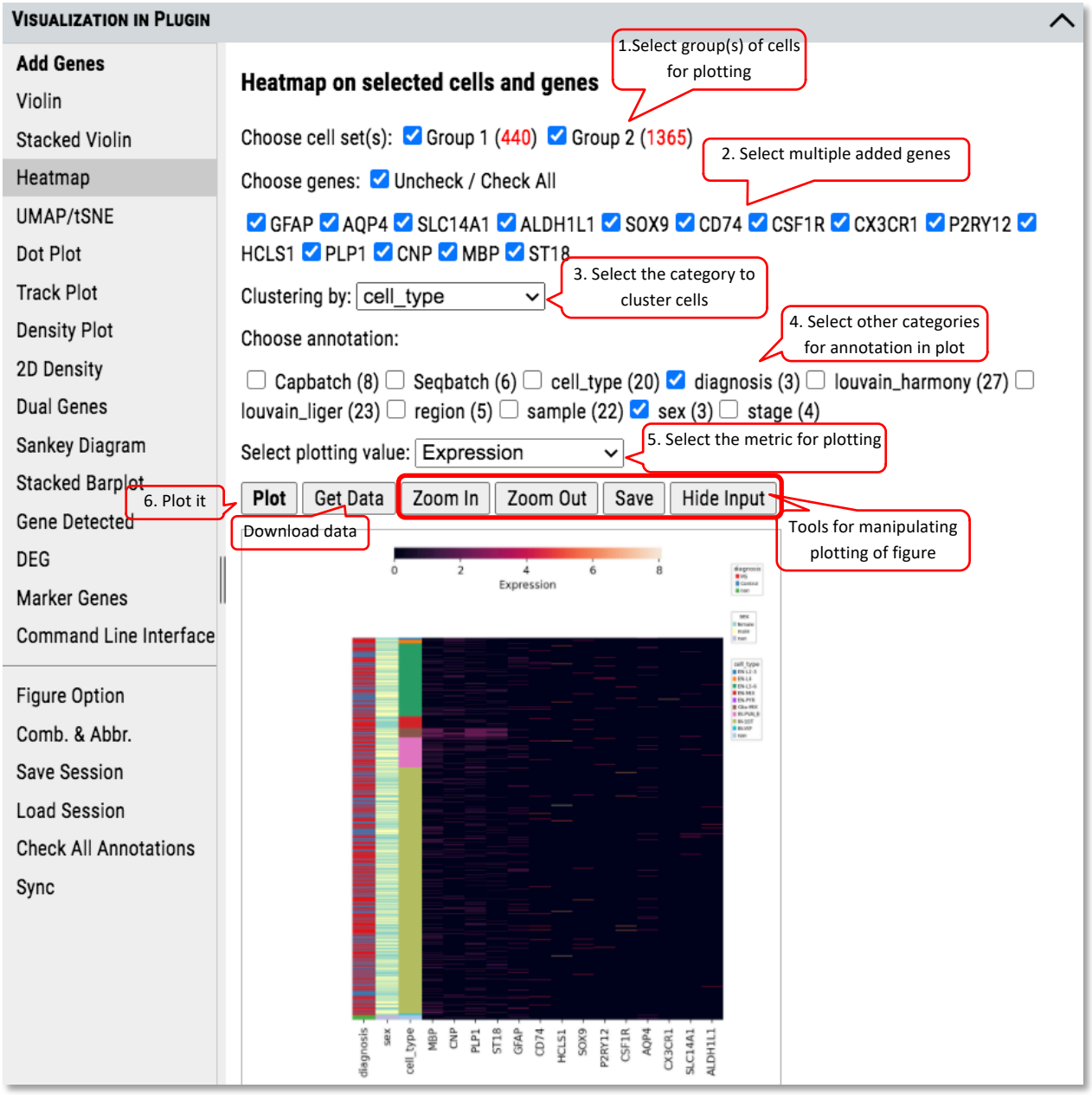


Fig. S9 | Heatmap of gene expression in cells grouped by annotations.

10. VIP – UMAP/tSNE

To plot the embedding of cells in the selected group(s). One of pre-computed and loaded embeddings can be selected.

User can color cells in the embedding plots by multiple annotations (e.g., cell_type, diagnosis).

Besides coloring cells by annotations, user can color cells based on gene expression level of selected genes in the embedding plots.

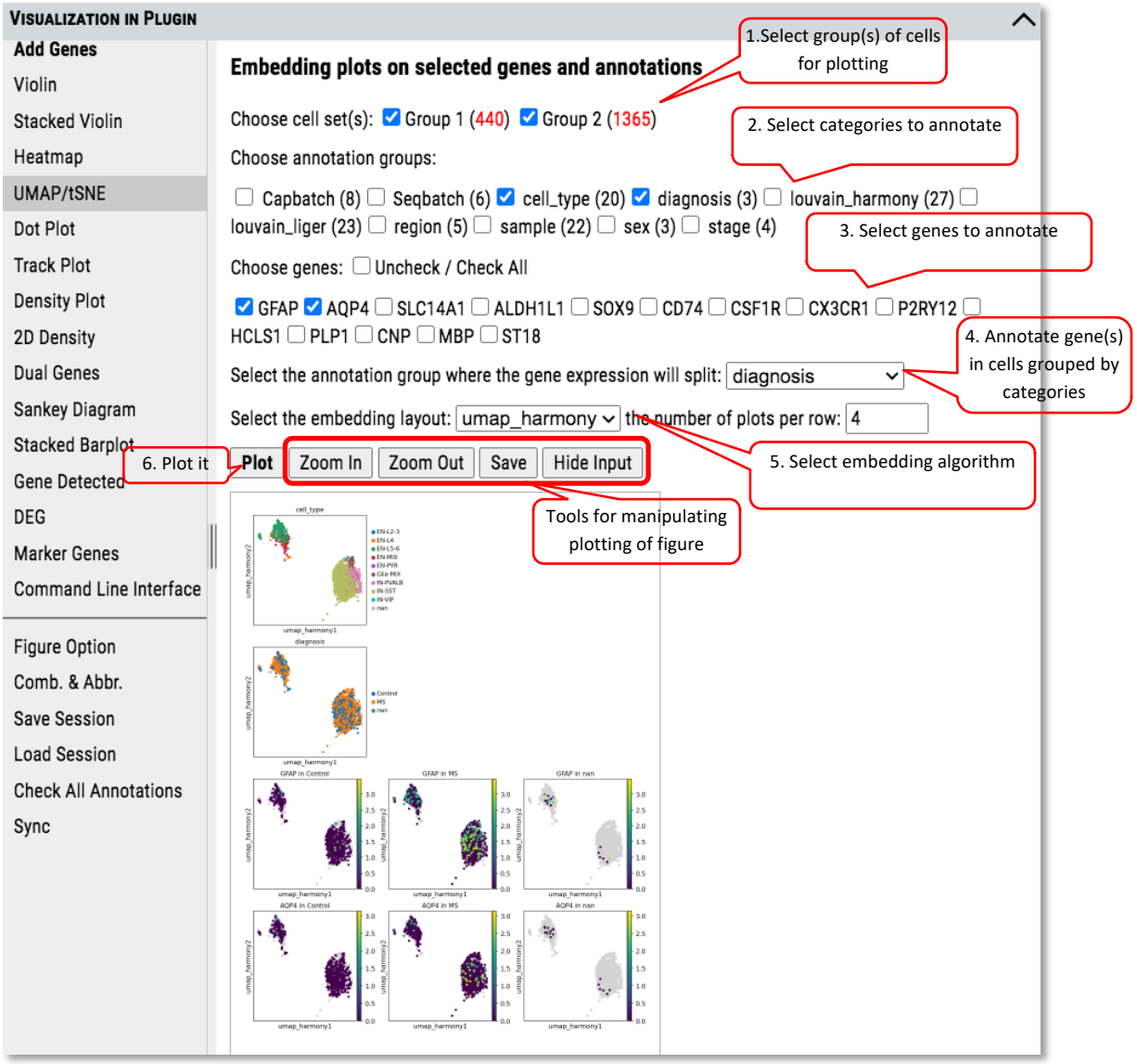


Fig. S10 | Embedding plotting of expression level of genes or gene set in the cells split by categories of an annotation.

11. VIP – Dot Plot

To show the fraction of cells (annotated by dot size) expressing a gene in each group and the averaged expression level of the gene (annotated by color intensity) in the group.

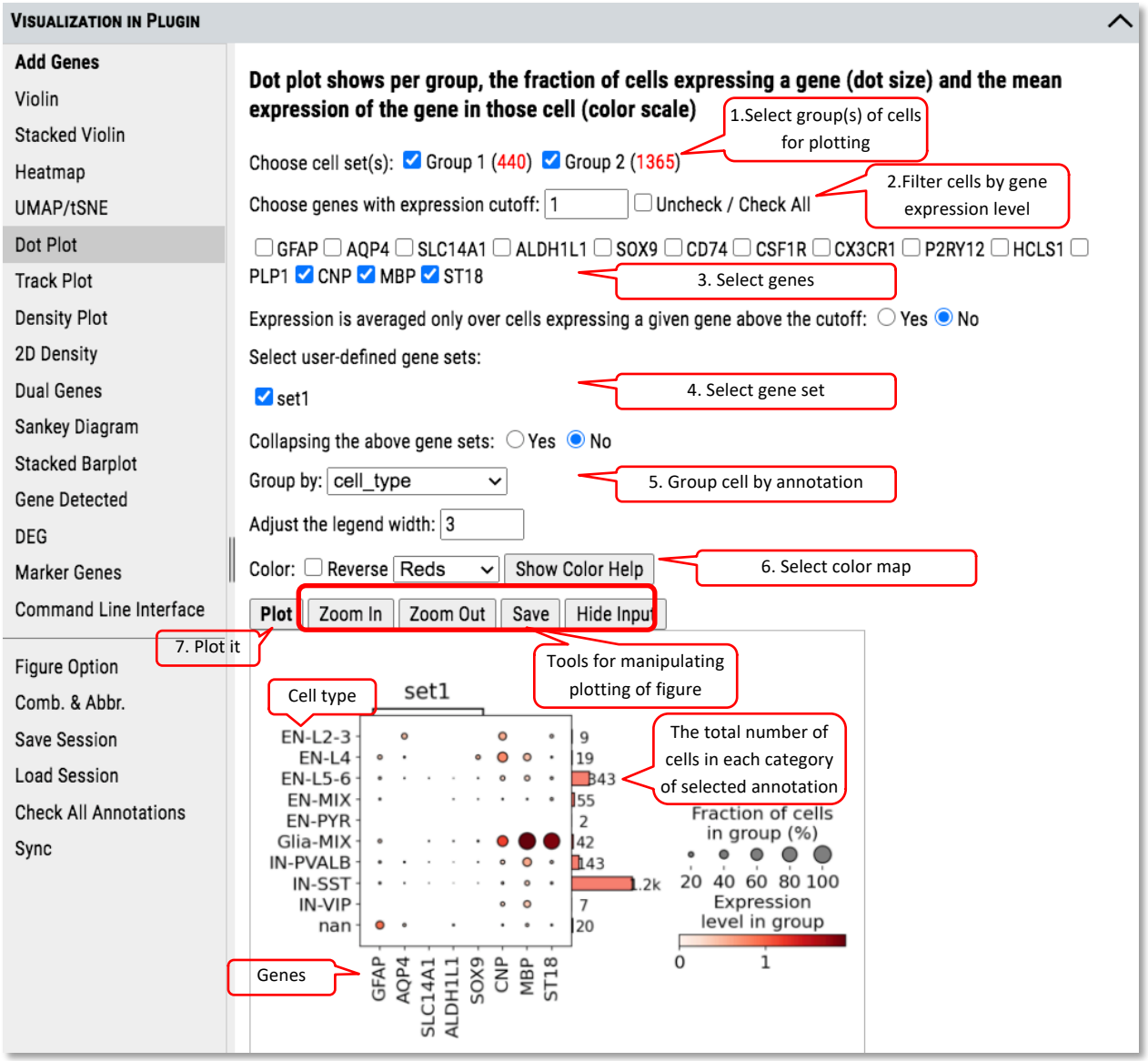


Fig. S11 | Dot plotting of the fraction of cells expressing genes above a cutoff in each categorie of the selected annotation.

Note: The number of cells represented by side bar chart are always numbers of cells distributed in each category of certain annotation without filtering. It will give accurate estimate of number of cells in each bubble in the plot. The use of the plot is only meaningful when the counts matrix contains zeros representing no gene counts. Its visualization does not work for scaled or corrected matrices in which zero counts had been replaced by other values, see https://scanpy-tutorials.readthedocs.io/en/multiomics/visualizing-marker-genes.html#Dot-plots.

12. VIP – Track Plot

To show the expression of gene(s) of individual cells as vertical lines grouped by the selected annotation on x-axis. Instead of a color scale, the gene expression is represented by height.

Fig. S12 | Track plotting of expression of genes or gene set in each category of the selected annotation. Gene expression levels are represented by the heights of vertical lines.

13. VIP – Density Plot

To show the density of gene(s) expression in the cells annotated by category in the selected group(s) of cells. A density plot is a representation of the distribution of a numeric variable. It uses a kernel density estimate to show the probability density function of the variable (see more). It is a smoothed version of the histogram and is used in the same concept.

The bandwidth defines how close to a value point the distance between two points must be to influence the estimation of the density at the point. A small bandwidth only considers the closest values, so the estimation is close to the data. A large bandwidth considers more points and gives a smoother estimation.

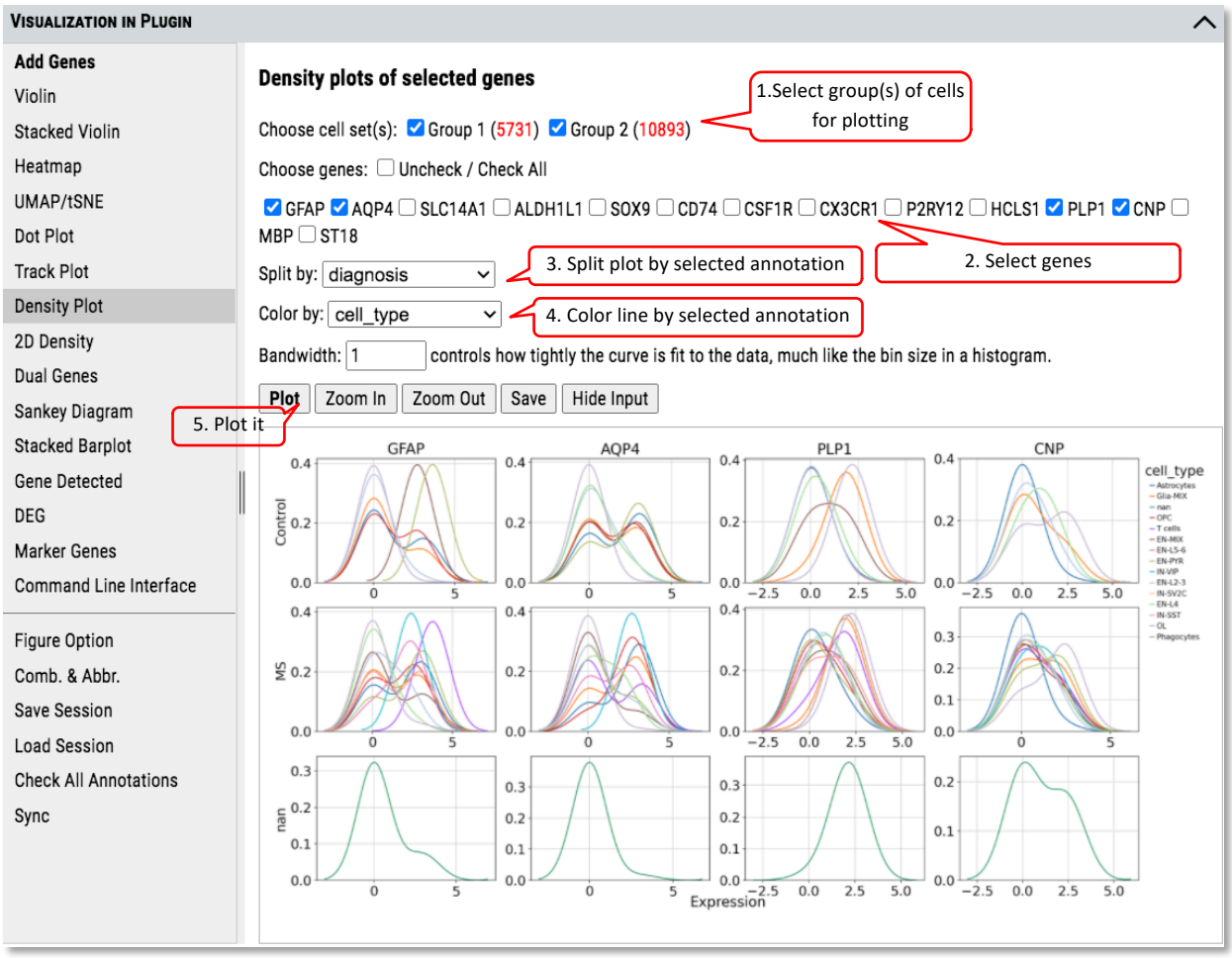


Fig. S13 | Density plotting of the expression of genes in each group split by one annotation while colored by another.

14. VIP – 2D Density Plot

Besides plotting of expression density of single gene, 2D density plot allows to explore the joint expression density of two genes in the cells expressing both genes above a cutoff.

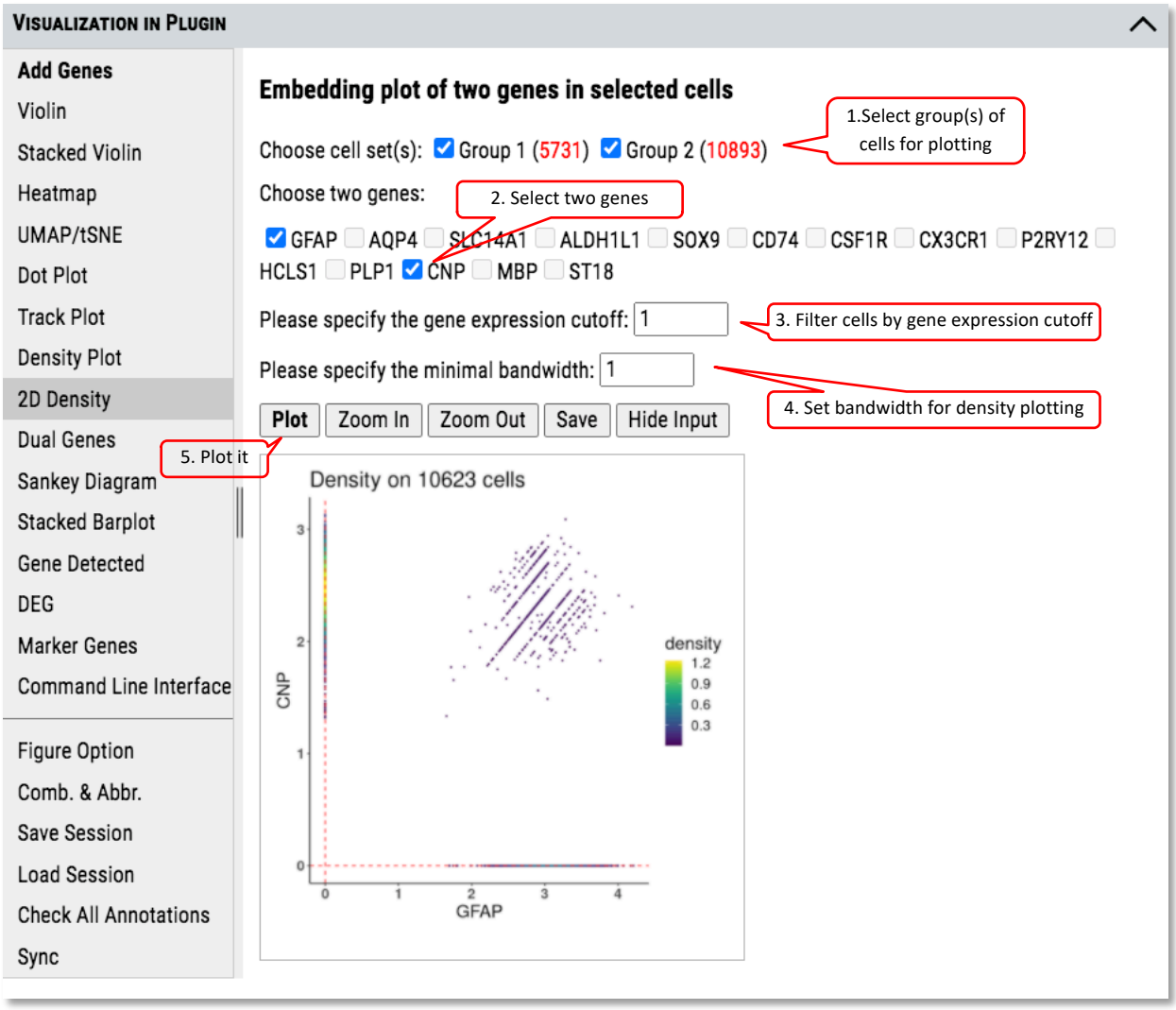


Fig. S14 | 2D Density plotting of expression of two genes in the selected cells.

15. VIP – Dual Genes

To view the relationship of expression levels of two genes in selected cells. It is based on the embedding plot of cells while coloring cells according to the expression levels of gene(s) in each cell.

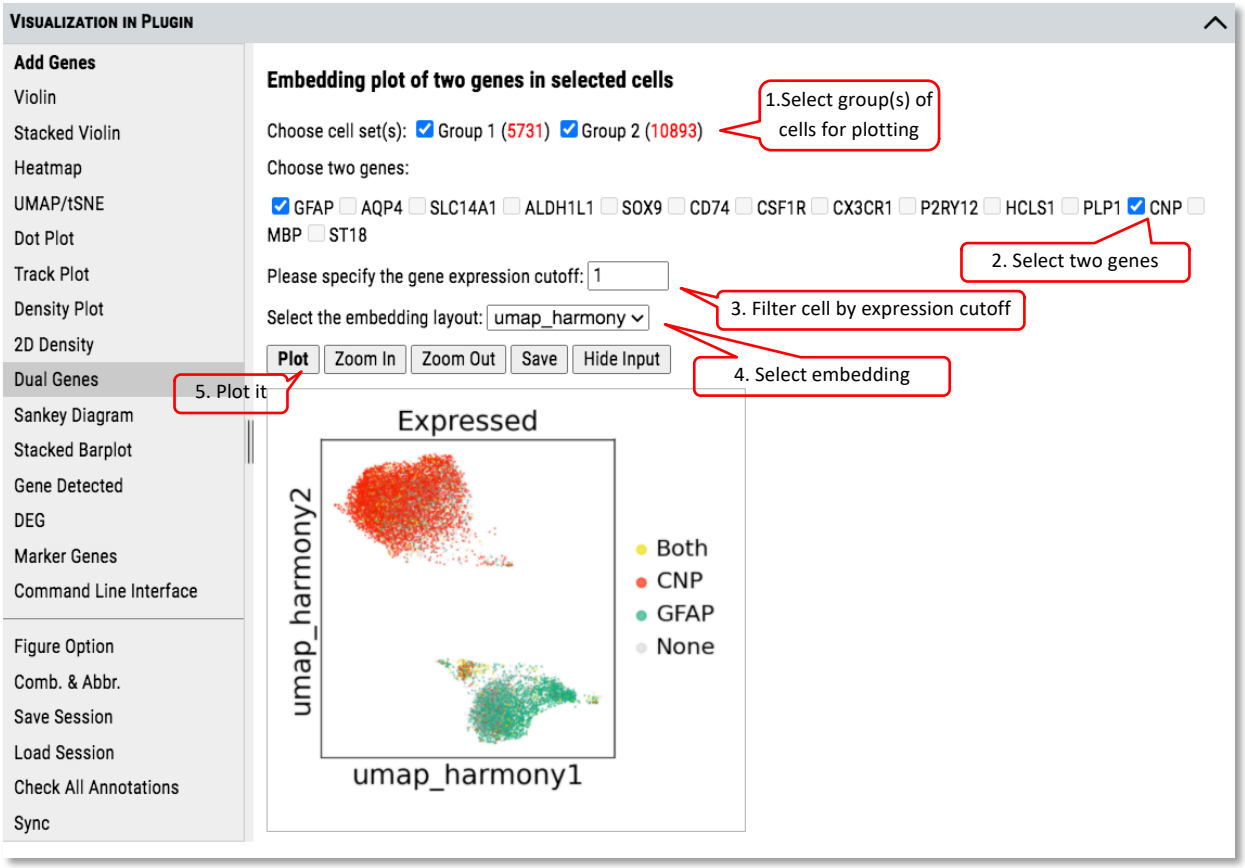


Fig. S15 | Embedding plotting of the expression of two genes in the selected cell group(s).

16. VIP – Sankey Diagram

Sankey diagram shows the flow of gene expression and annotations linked by cells. Gene expression is divided equally into bins so user can view distribution relationship between gene expression and annotations.

The diagram is also shown in an interactive way that user can change the layout by selecting several items (e.g., thin or thick on color bar, small or large space) from the panel. Also, user can drag these small boxes on the plot to get preferred layout and save it as high resolution SVG figure.

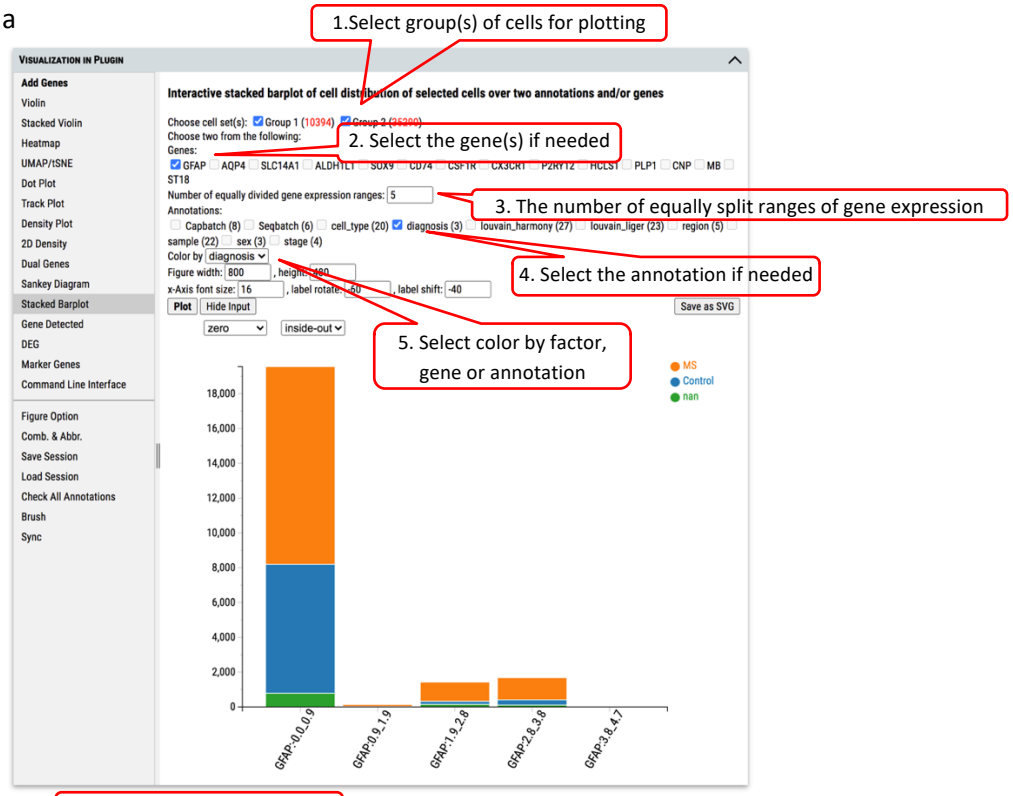
In addition, when you hover over mouse on a box, you can get detailed information about the source and target of flow.

Fig S16 | Sankey diagram provides quick and easy way to explore the inter-dependent relationship of variables.

17. VIP – Stacked Barplot

To show the distribution of cells among categories of an annotation and/or ranges of expression of a gene. Only two factors from annotations or genes can be chosen. The plot allows user to explore the distribution of cells in different views interactively as shown in Fig. S17b and Fig. S17c.

a



b

c

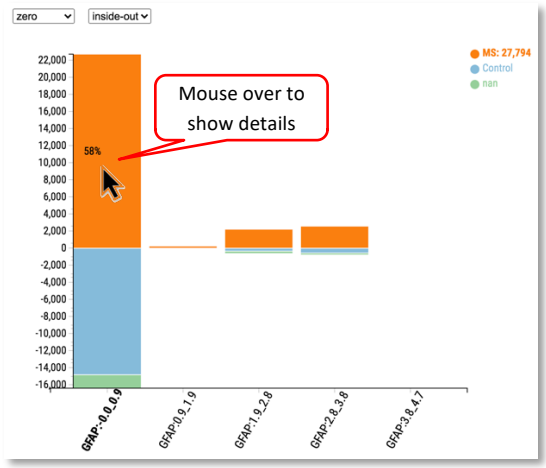


Fig. S17 | (a) The distribution of cells in the selected group(s) regarding categories of an annotation and expression ranges of a gene. (b) Expand view to show percentage instead of numbers of cells. (c) Mouse over a bar to show details. Y-axis is re-located for better view and the number of cells represented by the bar is also shown in legend.

18. VIP – Gene Detected

To show the number of genes expressed above the specified expression cut-off in the selected group(s) of cells.

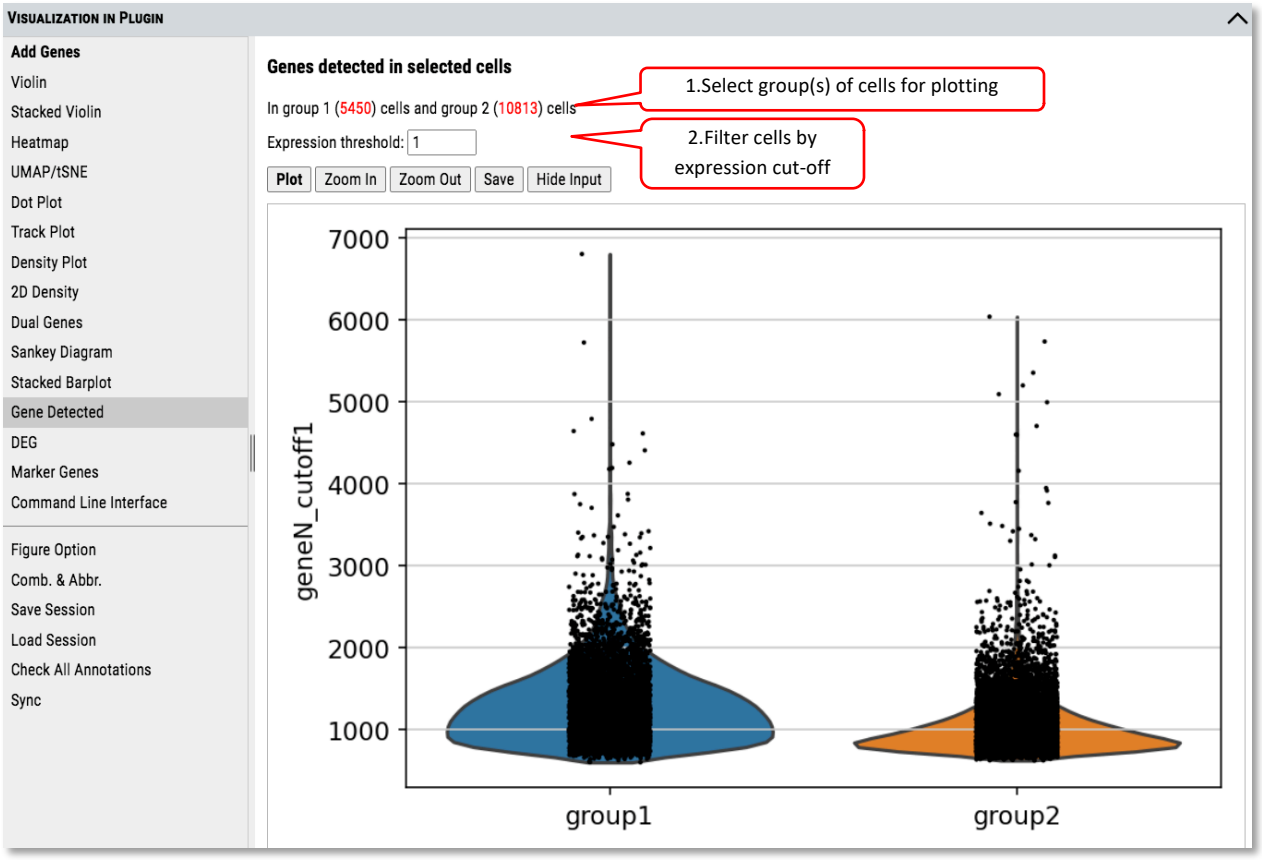


Fig. S18 | The number of genes with expression over the cut-off in the cells from the selected group(s) of cells.

19. VIP – DEG (Differential Expressed Genes)

Besides plotting functions, cellxgene VIP also provides differential analysis between two selected groups of cells to identify differential expressed genes.

Three differential analysis statistical test methods are provided including Welch's t-test, Wilcoxon rank test and Wald's test. The statistical test results are presented in a table format including log2 Fold change, p-value and padj value (i.e, FDR value). Please note, ***we provide users with simple test methods for quick exploration within the interactive framework. However, there would be covariates need to be considered in a proper statistical test. Please consult your stats experts for appropriate test by using the right test method and right model***.

Volcano plotting is also provided to show the log2FC vs. -log10(FDR) relationship for all genes. User can select the gene(s) from the pre-selected gene list to be highlighted with text in the volcano plot.

Fig. S19 | (a) DEG analysis between the selected group(s) and (b) Volcano plotting of identified DEGs.

Note: The data used by DEG is unscaled (please refer to description of the dataset to find out what preprocessing was done on the data). Scaling control in the Figure Option does not apply to DEG. The three methods: 'Welch's t-test' uses t-test (assuming underlining data with normal distributions) this uses **cellxgene** t-test implementation, 'Wilcoxon rank test' uses Wilcoxon rank-sum test (does not assume known distributions, non-parametric test) and 'Wald's test' uses Wald Chi-Squared test which is based on maximum likelihood. 'Wilcoxon rank test' and 'Wald's test' use diffxpy's implementation.

20. VIP – Marker Genes

This functional module allows user to identify marker genes in the selected group(s) (more than 2, if 2 groups, please use DEG) of cells by annotation categories.

Four methods are provided for detecting marker genes including logreg, t-test, Wilcoxon, and t-testoverest_var. For each identified marker gene, the gene name, scores (the z-score underlying the computation of a p-value for each gene for each group) and assigned group are listed in the output table.

In each annotation category, top ranked marker genes (this example shows top 2) will be plotted by score in comparison to the rest of the categories.

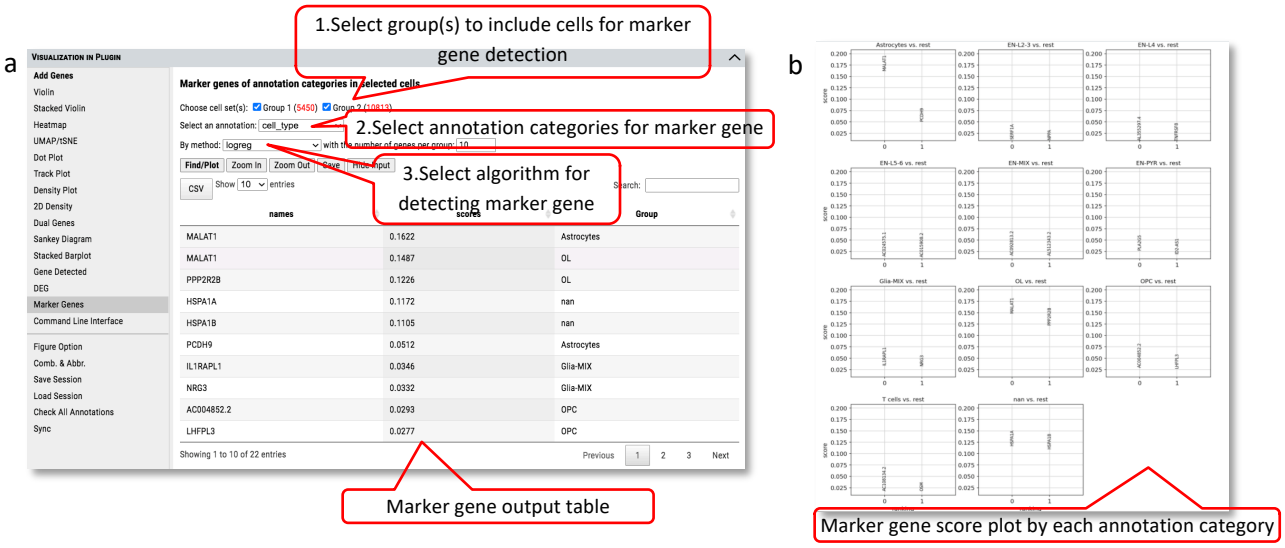


Fig. S20 | (a) Marker genes detection in the selected group(s) of cells regarding to the selected annotation categories and (b) plotting of identified marker genes by each category.

Note: The four methods implementations by calling scanpy.tl.rank_genes_groups function: 'logreg' uses logistic regression, 't-test' uses t-test, 'wilcoxon' uses Wilcoxon rank-sum, and 't-test_overestim_var' overestimates variance of each group.

21. VIP – Command Line Interface

Although cellxgene VIP provides a rich set of visualization modules as shown above, command line interface is also built to allow unlimited visualization and analytical capabilities by power user who know how to program in Python / R languages.

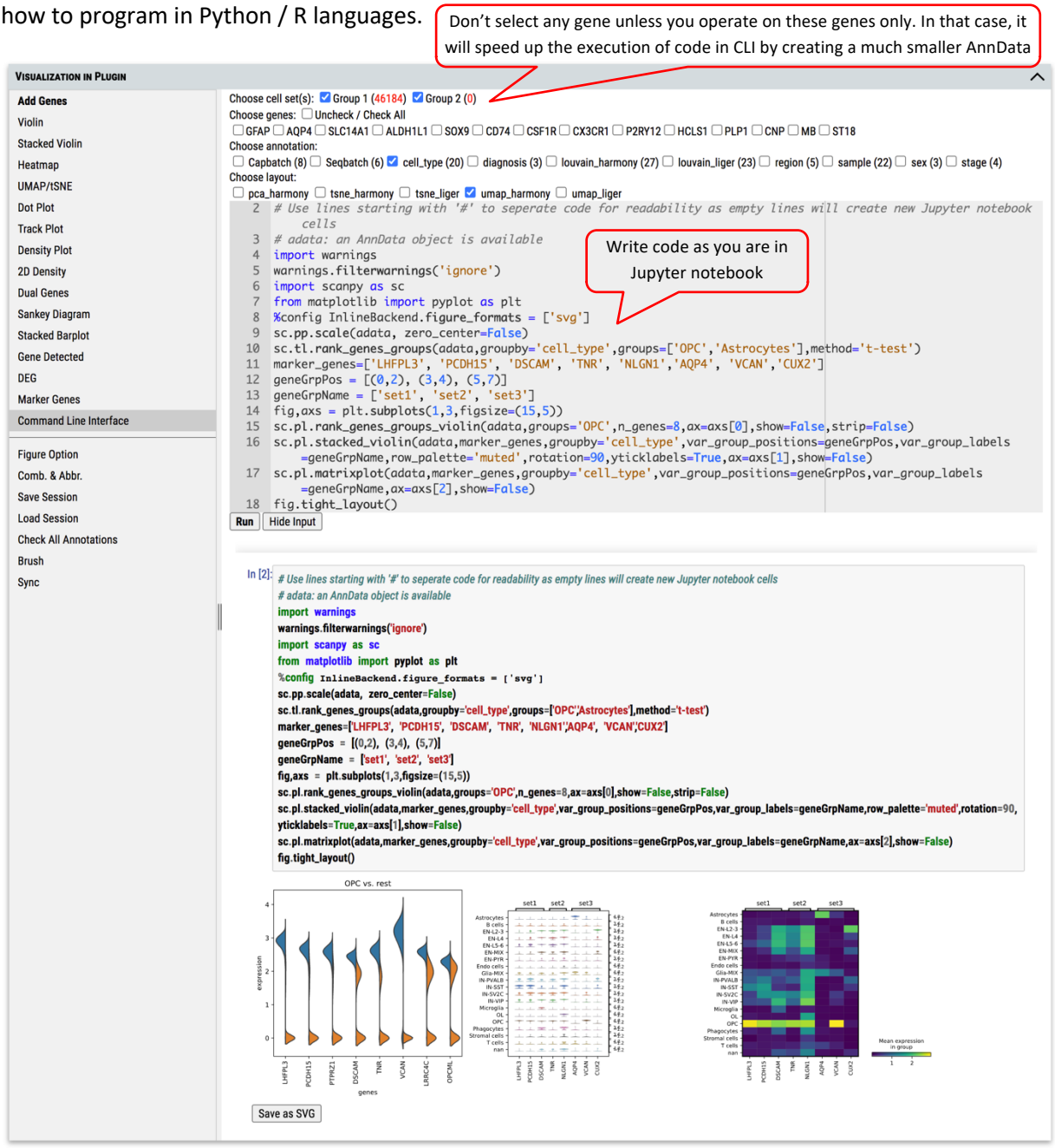


Fig. S21 | Command line interface for user to program for advanced plotting and statistical analysis.

Note: In CLI the AnnData (adata) object is available by default, and it is processed as 'Description' of the dataset states (i.e.: normalized and log transformed, but no scaled etc.). Settings in 'Figure Option' tab won't apply to CLI.

22. VIP – Comb. & Abbr.

User can combine multiple annotations to create combinatorial annotation to group cells in various of plotting, e.g., stacked violin and dot plot. Firstly, user 'uncheck all annotations' and, secondly go to the annotation panel in main window to select annotation categories to be combined, e.g., diagnosis (Control and MS) combined with sex (female and male). After clicking on 'Create' button, all possible combinatorial names will be listed and 'Custom_combine' will be automatically available as an option in 'Group by' drop down menu of many plotting functions.

User can also rename each annotation by creating abbreviations to shorten axis labels in figures.

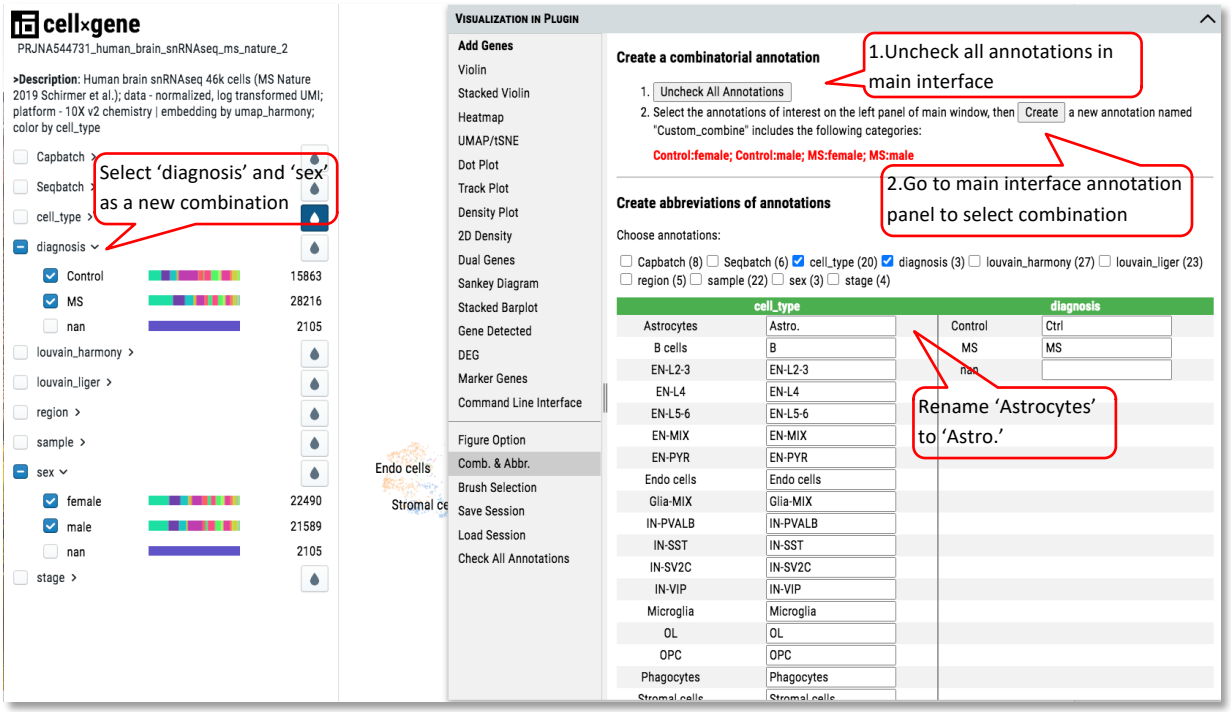


Fig. S22 | Comb. & Abbr. function allows user to create new annotation by combining multiple annotations and abbreviations to shorten axis labels in figures especially when custom combinatorial names are used.

23. VIP – Other Functions

There are other convenient functions available to user, such as ‘Save’ or ‘Load’ session, ‘Check All Annotations’ and ‘Brush’.

‘Save’ or ‘Load’ session are used to save the current cell selections and parameter settings to text file or load previously saved session file in the tool for visualization.

‘Check All Annotations’ is used to check all categorical selection boxes of annotations on the left panel.

‘Brush’ is to display exactly these selected ranges from histograms of variables on the right panel in a nice table that is not available in original cellxgene.

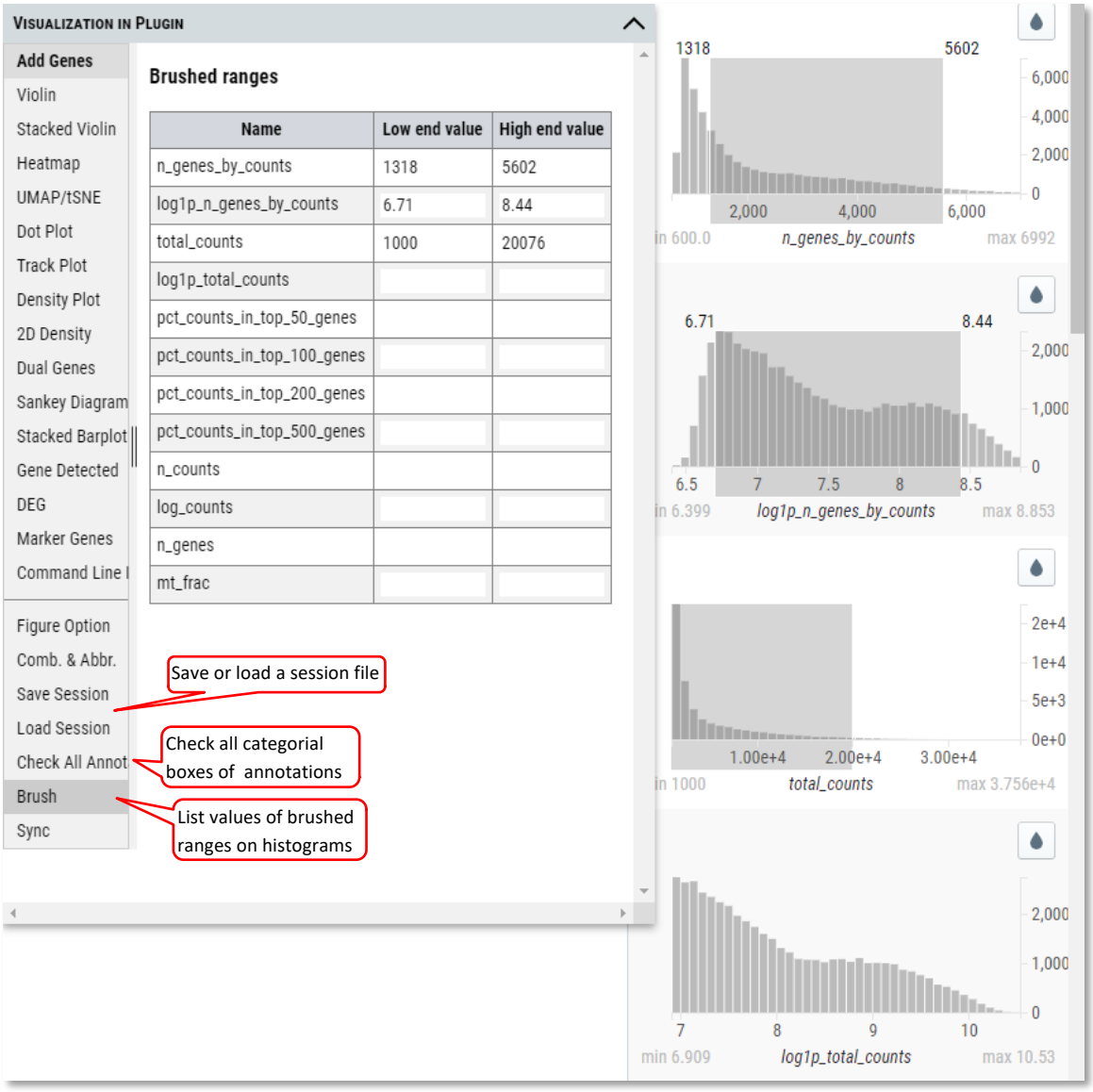


Fig. S23 | Other functions allow user to ‘Save’ or ‘Load’ session information, ‘Check All Annotations’ and show values of brushed ranges on histograms.

Sup. Method 1 – Client-side Integration by a jsPanel Window (VIP)

Following section in config.sh file.

```
read -d '' insertL << EOF
<script src="static/jquery.min.js"></script>
<link href='static/jspanel/dist/jspanel.css' rel='stylesheet'>
<script src='static/jspanel/dist/jspanel.js'></script>
<script src='static/jspanel/dist/extensions/modal/jspanel.modal.js'></script>
<script src='static/jspanel/dist/extensions/tooltip/jspanel.tooltip.js'></script>
<script src='static/jspanel/dist/extensions/hint/jspanel.hint.js'></script>
<script src='static/jspanel/dist/extensions/layout/jspanel.layout.js'></script>
<script src='static/jspanel/dist/extensions/contextmenu/jspanel.contextmenu.js'></script>
<script src='static/jspanel/dist/extensions/dock/jspanel.dock.js'></script>
<script>
    // execute JavaScript code in panel content
    var setInnerHTML = function(elm, html) {
        elm.innerHTML = html;
        Array.from(elm.querySelectorAll('script')).forEach( oldScript => {
            const newScript = document.createElement('script');
            Array.from(oldScript.attributes)
            .forEach( attr => newScript.setAttribute(attr.name, attr.value) );
            newScript.appendChild(document.createTextNode(oldScript.innerHTML));
            oldScript.parentNode.replaceChild(newScript, oldScript);
        });
    }
    var plotPanel = jsPanel.create({
        panelSize: '190 0',
        position: 'left-top 160 6',
        dragit: { containment: [-10, -2000, -4000, -2000] }, // set dragging range of VIP window
        boxShadow: 1,
        border: "solid #D4DBDE thin",
        contentOverflow: 'scroll scroll', // adding scrolling bars
        headerControls:{
          close: 'remove',
          minimize: 'remove',
          maximize: 'remove'
        },
        headerTitle: function () {return '<strong>Visualization in Plugin</strong>'},
        contentAjax: {
            url: 'static/interface.html',
            done: function (panel) {
                  setInnerHTML(panel.content, this.responseText);
            }
        },
        onwindowresize: function(event, panel) {
            var jptop = parseInt(this.currentData.top);
            var jpleft = parseInt(this.currentData.left);
            if (jptop<-10 || window.innerHeight-jptop<10 || window.innerWidth-jpleft<10 ||
jpleft+parseInt(this.currentData.width)<10) {
                this.reposition("left-top 160 6");
            }
        },
        onunsmallified: function (panel, status) {
            this.reposition('center-top -370 180');
            this.resize({ width: 740, height: function() { return Math.min(480, window.innerHeight*0.6);} });
        },
        onsmallified: function (panel, status) {
            this.reposition('left-top 160 6');
            this.style.width = '190px';
        }
    }).smallify();
    plotPanel.headerbar.style.background = "#D4DBDE";
</script>
EOF
insertL=$(sed -e 's/[&\\/]/\\&/g; s/|/\\|/g; s/$/\\/;' -e '$s/\\$//' <<<"$insertL")
sed -i "s|<div id=\"root\"></div>|$insertL\n&|" "cellxgene/client/index_template.html"
```

All functional VIP HTML and JavaScript code will be in “interface.html” that is independent of cellxgene code base.

Sup. Method 2 – Server-side Integration

Following section in config.sh file.

```
echo '
from server.app.biogenInterface import route
@webbp.route("/biogen", methods=["POST"])
def biogen():
    return route(request.data,current_app.app_config)' >> cellxgene/server/app/app.py
.
.
.

strPath="$(python -c 'import site; print(site.getsitepackages())')"
strPath=${strPath//"['"/}
strPath=${strPath//"']"/}
strweb="${strPath}/server/common/web/static/."
echo $strweb
cp interface.html $strweb
cp jquery.min.js $strweb
cp color_map.png $strweb

cp -R DataTables $strweb
cp -R jspanel $strweb

cp cellxgene/server/test/decode_fbs.py $strPath/server/app/.
cp VIPInterface.py $strPath/server/app/.
```

Sup. Method 3 – Communication between VIP and cellxgene web GUI

Cellxgene client utilizes React Redux that is the official React binding for Redux. It lets your React components read data from a Redux store, and dispatch actions to the store to update data.

So, this line of code is appended to the end of client/src/reducers/index.js of cellxgene source code to expose the store to the browser.

```
window.store = store;
```

By doing this, Redux store holding client data and user selections are visible to VIP to access variables and dispatch actions to control cellxgene user interface. For example,

1. Unselect / select a feature. GUI is refreshed automatically after dispatching.

```
window.store.dispatch({type: "categorical metadata filter deselect", metadataField: "louvain",
categoryIndex: 5})
window.store.dispatch({type: "categorical metadata filter select", metadataField: "louvain",
categoryIndex: 5})
```

2. Get state of just finished action and synchronize gene input and cell selections from main window to VIP if corresponding action was performed.

```
const unsubscribe = window.store.subscribe(() => {
  if (window.store.getState()["@@undoable/filterState"].prevAction) {
    actionType = window.store.getState()["@@undoable/filterState"].prevAction.type;
    if (actionType.includes("user defined gene success") ||
        actionType.includes("store current cell selection as differential set")) {
      sync();
    }
  }
});
```

Sup. Method 4 – Diffxpy Integration

This is the sample pseudocode, please see VIPInterface.py for actual implementation.

```
import scanpy as sc
import pandas as pd
import diffxpy.api as de
import server.app.app as app
# set 1 of cells as cell1; set 2 of cells as cell2

with app.get_data_adaptor() as data_adaptor:
  X1 = data_adaptor.data.X[cell1]
  X2 = data_adaptor.data.X[cell2]

adata = sc.AnnData(pd.concat([X1,X2]),pd.DataFrame(['grp1' for i in range(X1.shape[0])]+['grp2' for i
    in range(X2.shape[0])],columns=['comGrp']))
deg = de.test.two_sample(adata,'comGrp').summary()
#deg is a dataframe contains the folloing columns ['gene','log2fc','pval','qval']
```

Sup. Method 5 – Create h5ad file from Seurat object

First, export the following from Seurat object in R: **expression matrix (assume normalized), metadata and coordinates (pca, tsne, umap) as separate txt files**.

Next in Python, create an AnnData object from 10x (*scanpy.read_h5ad function*) as a starting point. Then replace the expression matrix, meta data and coordinates as following, a h5ad file would be generated.

Python code:

```
import sys
import scanpy as sc
import pandas as pd
import numpy as np
import seaborn as sns
from numpy import ndarray, unique
from scipy.sparse.csc import csc_matrix

adata= sc.read_h5ad("previous generated .h5ad")

# read clustering res
xpca = pd.read_csv(“./data/harmony_clustered.h5ad.pca_coordinates.txt", sep='\t', encoding='utf-8')
xtsne = pd.read_csv(“./data/harmony_clustered.h5ad.tsne_coordinates.txt", sep='\t', encoding='utf-8')
xumap = pd.read_csv(“./data/harmony_clustered.h5ad.umap_coordinates.txt", sep='\t', encoding='utf-8')
xobs = pd.read_csv(“./data/harmony_clustered.h5ad.meta_data.txt", sep='\t', encoding='utf-8')

xpca.set_index('index', inplace=True)
xtsne.set_index('index', inplace=True)
xumap.set_index('index', inplace=True)
xobs.set_index('index', inplace=True)

adata.obsm['X_pca'] = np.array(xpca.loc[adataRaw.obs.index])

adata.obsm['X_tsne'] = np.array(xtsne.loc[adataRaw.obs.index])
adata.obsm['X_umap'] = np.array(xumap.loc[adataRaw.obs.index])
adata.obs = xobs.loc[adataRaw.obs.index] # this is a pandas dataframe

# read in expression matrix as numpy.ndarray as following:
exp_mat = np.loadtxt(fname =”expression matrix .txt")
adata.X = exp_mat
# convert dense matrix into sparse matrix to save storage space and memory usage
adata.X = csc_matrix(adata.X)_matrix
```

```
# add short description and initial graph settings. "|" and "by" are delimiters for VIP to parse the initial settings. Please follow the same rule for your own h5ad files.

adata.obs['>Description'] = ['Human brain snRNAseq 46k cells (MS Nature 2019 Schirmer et al.); data - normalized, log transformed and scaled UMI; platform - 10X v2 chemistry | embedding by umap; color by cell_type']*adata.n_obs

# Then last step to save h5ad:
adata.write_h5ad("final output.h5ad")
```

Helpful tips

Handle nulls in categorical annotation

Such nan's in categorical annotation would cause trouble in VIP because it cannot be converted to string. Here is how to handle it, let's call the annotation X_annotation:

```
# Cast to str from categorical

adata.obs = adata.obs.astype({'X_annotation':'str'})

# replace all of nan by 'nan'

adata.obs["X_annotation "][adata.obs["X_annotation "].isnull()] = 'nan'
```

Display full traceback stack for debugging in VIP

It follows the global setting. Please set "—verbose" to launch cellxgene server.

Pitfall of using special characters

In the mode which allows user to create manual annotation in cellxgene, user should try to avoid using hyphen ("-") in name label.It would cause client-side issue. Please try to use underscores.

Potential use for bulk or pseudo bulk sample dataset

Once the data matrix is replaced by sample x gene matrix, cellxgene VIP framework can handle regular bulk / pseudobulk RNAseq datasets. Simply replace "cells" by "samples". All plotting functions would still work.