DOI: 10.1002/imt2.115

CORRESPONDENCE

PyComplexHeatmap: A Python package to visualize multimodal genomics data

INTRODUCTION

Advancements in science and technology in the field of biology, including high-throughput sequencing, have led to an increase in sample size and sequencing coverage. To handle the ever-growing data volume, numerous Python software and algorithms have been developed. For example, SCANPY [1], EpiScanpy [2], and scVelo [3] are useful for processing and analyzing single-cell sequencing data, while PyDESeq2 [4] and GSEApy [5] facilitate differential expression gene analysis and gene set enrichment analysis, respectively. Squidpy [6] is a recently introduced package for spatial single-cell analysis. Furthermore, Python is widely employed for machine learning and deep learning. An increasing number of Python packages are being developed to address specific challenges, such as imputing missing values in single-cell sequencing data [7] and integrating single-cell omics data [8] using deep learning.

Despite the popularity of Python in machine learning, deep learning, and big data processing, there is currently no powerful package for generating complex heatmaps similar in caliber to R's *ComplexHeatmap* [9, 10]. This has become a pressing issue for researchers who utilize Python for data analysis, machine learning, and deep learning. Consequently, there is an urgent need for a Python package capable of generating highly complex heatmaps.

To fill this gap in Python, we developed *PyComplexHeatmap*, a Python package that enables users to easily visualize multidimensional biological data. We have added several new features to enhance user-friendliness while accommodating the Python environment. *PyComplexHeatmap* possesses multiple features that are not included in the current state-of-the-art Python heatmap library *Seaborn* [11]. With *PyComplexHeatmap*, users can: (i) Plot various types of rows or columns annotations (e.g., simple heatmap annotation, barplot annotation, boxplot annotation, scatterplot annotation, and label annotations) separately or together with the main heatmap, with the ability to add group labels on the top of simple heatmap annotations by simply turning on a parameter. (ii) Add label annotations for specific rows or columns quickly and evenly distribute the labels throughout the axis without extra manual editing. (iii) Easily choose different palettes using python matplotlib [12] build-in colormap, custom colormap, or colors mapping dict. (iv) Combine two or more heatmaps horizontally or vertically to create a joint figure. (v) Visualize up to five variables in the data using a dot heatmap.

RESULT

Implementation

PyComplexHeatmap is a Python package that is built on a few essential Python libraries, including matplotlib [12], pandas (https://github.com/pandas-dev/pandas), scipy [13], and numpy [14]. As illustrated in Figure 1A, the layout of PyComplexHeatmap includes the column annotations (top and bottom), row annotations (left and right), main heatmap (optionally split up into multiple rows and columns), and figure legend (optional). The package comprises several primary classes, including:

ClusterMapPlotter: This class enables the user to plot a basic heatmap and perform row and column clustering using different linkage methods and metrics. Additionally, users can add multiple annotations to the main heatmap's top, bottom, left, and right sides by providing a HeatmapAnnotation object as the parameter. Rows or columns can also be split according to metadata or hierarchical clustering.

HeatmapAnnotation: This class can be passed to [left/right/top/bottom]_annotation in ClusterMapPlotter or DotClustermapPlotter. HeatmapAnnotations allows users to add boxplot, barplot, and scatter plot annotations

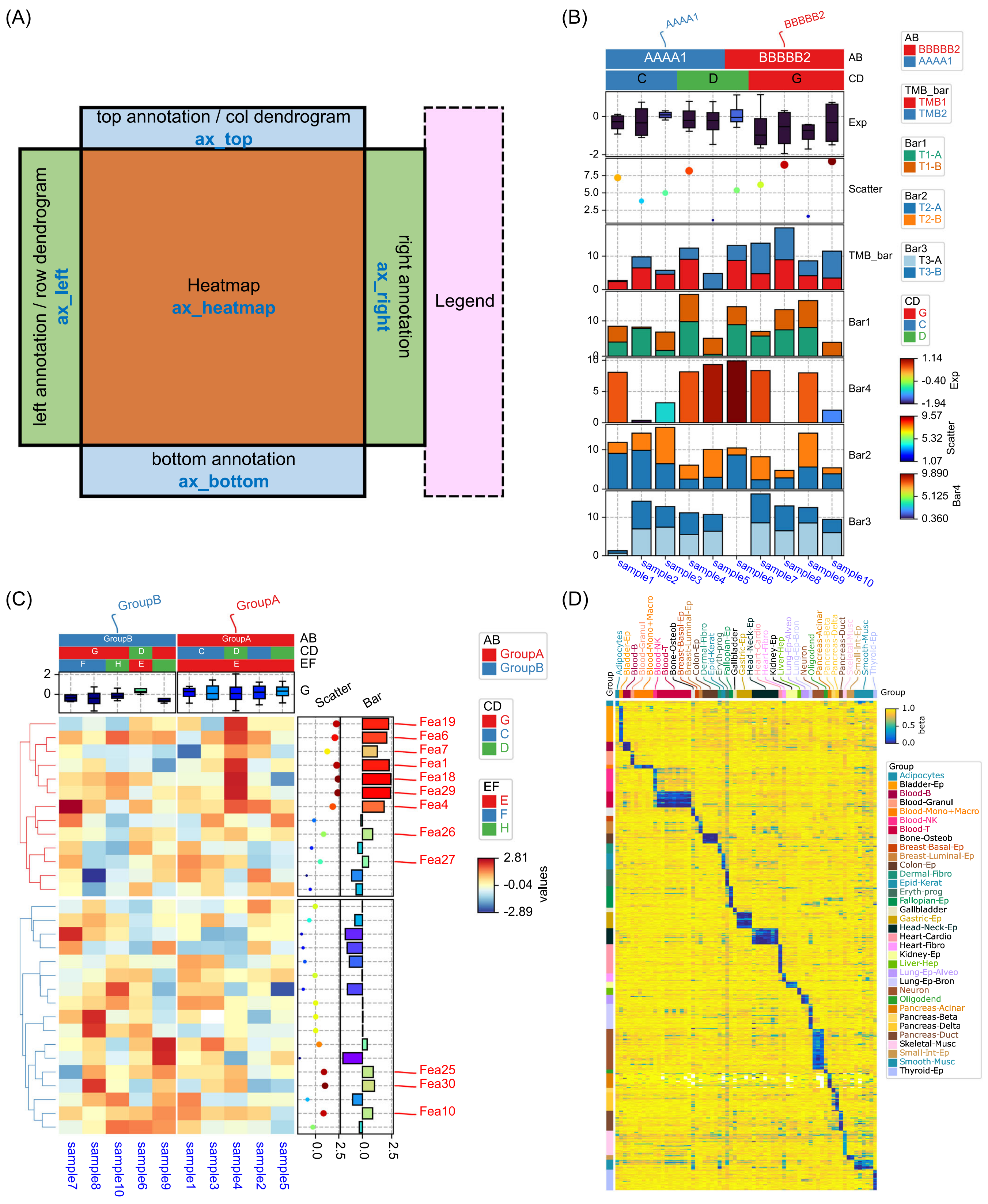

FIGURE 1 Layout and example heatmaps generated by PyComplexHeatmap. (A) The layout of PyComplexHeatmap. (B) Heatmap annotation is plotted separately to visualize the simulated categorical and continuous data simultaneously. (C) Heatmap annotation is plotted alongside the main heatmap. In addition to rows and columns annotations, a label annotation is included to emphasize the interested rows. (D) Heatmap of cell type-specific CpG methylation markers in different groups from the Loyfer2023 data set.

to the heatmap. Label annotations can also be added, and the anno_label merges the labels of the samples belonging to the same group and distributes them evenly throughout the axis without overlapping. Custom annotation classes can be added, and users can include as many kinds of annotations as desired.

```
col_ha = HeatmapAnnotation(...,axis=1,
cmap='set1', label_kws = {...}, ...)

row_ha = HeatmapAnnotation(...,axis=0,
cmap='set1', label_kws = {...}, ...)
```

The main heatmap can be accompanied by heatmap annotations, which can either be plotted alongside the main heatmap or separately. For instance, PyComplex-Heatmap can be used to visualize categorical and continuous data simultaneously in Figure 1B by incorporating heatmap annotations. Anno_barplot can automatically determine whether to generate a bar plot or a stacked bar plot depending on the structure of the input data frame.

DotClustermapPlotter: A class inherited from ClusterMapPlotter to plot dot heatmap using a similar input style as seaborn. The user can visualize up to five-dimensional biological data, including the *x*- and *y*-axis, the size, colormap, marker, and hue of the dot by providing a pandas data frame and other parameters.

oncoPrintPlotter: A class to plot oncoPrint plot, which was also inherited from ClusterMapPlotter. In addition to OncoPrint, users can also visualize any other categorical data using oncoPrintPlotter.

Composite: The composite function is used to combine two or more ClusterMapPlotter objects horizontally or vertically as a joint figure without any manual editing.

```
composite(cmlist = [cm1, cm2], main=0, ax=
None, axis=1,...)
```

Applications

The heatmap in Figure 1C displays annotations for both rows (anno_scatterplot, anno_barplot, and anno_label) and columns (anno_simple and anno_boxplot) using simulated datasets consisting of randomly generated normal distributed data. The samples were divided into two groups using the parameter col_split = df_col.AB, while rows were segregated into two groups based on the clustering tree structure generated by scipy.cluster.hierarchy.fcluster, using row_split = 2. On the right side of the heatmap, the scatter plot, and barplot annotations exhibit the data for sample 4, with labels displayed for selected features where sample 4 > 0.5.

Figure 1D illustrates the methylation signatures specific to different cell types. The CpG signatures and data were obtained from Loyfer et al. [16]. The label annotations on the top of this heatmap were added using anno_label as part of the column annotation. Labels from different samples belonging to the same cell group were merged and distributed throughout the *x*-axis with a simple line of code.

```
anno_label(df_col.Group, merge=True, rota-
tion=90, extend=True, colors=col_colors_
dict, ...)
```

In Figure 2A, we demonstrate the influence of strain-specific SNP on mouse array DNA methylation reading [15]. To represent the three wild mouse strains, all samples were divided into three groups using strain as the column split criterion. Probes with SNP were further divided into multiple groups based on the combination of target and group as rows split. By setting add_text = True in the anno_simple, strain names can be added easily to the top of the simple annotation. The left panel illustrates the SNP pattern in different probes across various strains, while the right panel displays the actual DNA methylation beta values for the corresponding probes and samples. We generated two panels separately using ClusterMapPlotter and combined them using the composite function to form a joint figure.

Figure 2B utilized a dot cluster map (DotClustermap-Plotter) to illustrate the correlation matrix of six CpG modules identified through the KnowYourCG package (https://www.bioconductor.org/packages/release/bioc/vignettes/sesame/inst/doc/KYCG.html). The size of the dots on the heatmap represented the correlation coefficient (*r*), and a variety of annotations were incorporated. Furthermore, the dot heatmap was divided into separate clusters based on the CpG modules. The different modules were enriched in distinct ChromHMM categories and exhibited varying mean DNA methylation levels, as demonstrated in the top annotation.

Using the PyComplexHeatmap layout as a foundation, we can create the oncoPrint heatmap, which is derived from the ClusterMapPlotter with some adjustments. Figure 2C illustrates the visualization of TCGA (The Cancer Genome Atlas) lung adenocarcinoma carcinoma variants profile through the oncoPrint heatmap. Like the DotClustermapPlotter and ClusterMapPlotter, various annotations can be added to the oncoPrint and the heatmap can also be split. Furthermore, it is possible to merge two or more oncoPrint heatmaps horizontally or vertically.

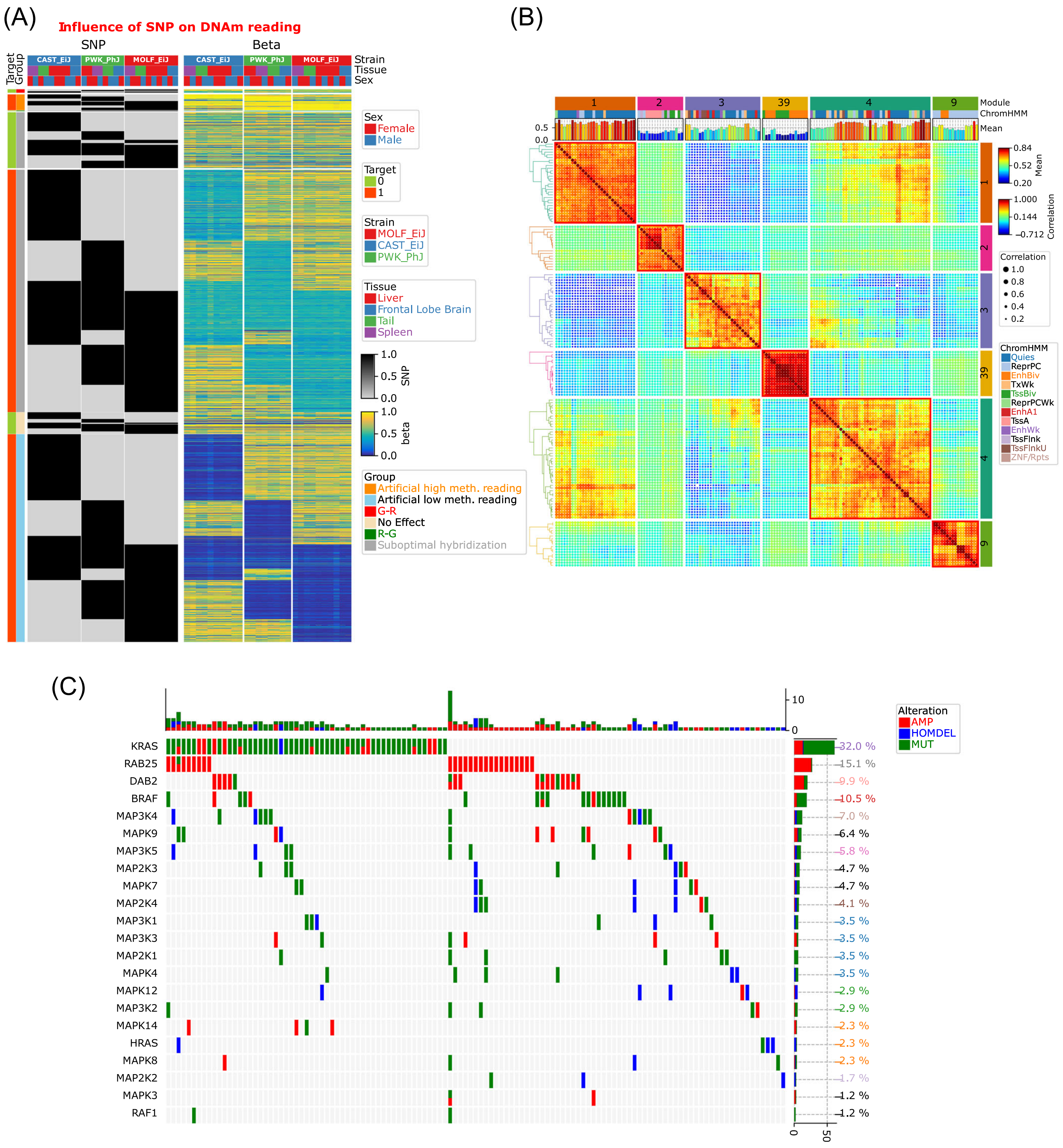

FIGURE 2 Illustration of heatmaps composition, dot heatmap, and oncoPrint. (A) The influence of mouse strain-specific SNP on DNA methylation reading of MM285 array. Two heatmaps were combined into a joint figure. (B) Dot heatmap showing the correlation matrix for CpGs coming from different modules. (C) OncoPrint of the TCGA (The Cancer Genome Atlas) lung adenocarcinoma carcinoma variants profile.

We developed this package as an initial step towards creating complex heatmaps in Python, and we believe that with the help of the open-source community on Github, it can be enhanced further. For more examples, please refer to the documentation site: https://dingwb.github.io/PyComplexHeatmap.

DISCUSSION

As single-cell sequencing sample sizes continue to increase, more Python packages are emerging to aid in the processing and analysis of single-cell genomic and epigenomic datasets. This manuscript introduces

TABLE 1 Comparison of the processing time and maximal memory usage for PyComplexHeatmap and ComplexHeatmap.

Package name	Processing time (s)	Memory (kb)
ComplexHeatmap	40.21	3,366,768
PyComplexHeatmap	22.57	1,037,944

PyComplexHeatmap, a Python package that simplifies and accelerates the generation of complex heatmaps in Python. Our package was compared to state-of-the-art data visualization packages, including matplotlib, seaborn, and ComplexHeatmap (Supporting Information: Table S1). PyComplexHeatmap currently boasts the most robust heatmap visualization capabilities in Python, with most of the features not available in other Python packages. In comparison to the R package ComplexHeatmap, many features in PyComplexHeatmap, such as automatically adding text above annotations, automatically distributing annotation labels, and generating dot heatmaps, are easier to use and can be produced automatically. When comparing running time and memory usage, PyComplexHeatmap performs better than R ComplexHeatmap when visualizing the same data set using the same clustering method and metric, running faster and consuming only one-third of the memory (Table 1). The data set and code utilized to compare these two packages can be found at https://github.com/DingWB/PyComplexHeatmap/tree/main/comparison.

While PyComplexHeatmap has some advantages, it is not as versatile as R ComplexHeatmap. Our package provides a framework for developing various heatmap visualization functions, and we anticipate that further efforts will be required by us and the Python community to optimize and update the package, including supporting more flexible input data format, implementing parallel computing and plotting, integrating more clustering methods, accommodating polar axes, providing interactive visualization, and integrating with other Python packages such as Scanpy, Django in the future.

CONCLUSION

We introduced PyComplexHeatmap, a versatile and user-friendly Python package, to fill the multidimensional data visualization gap in the Python-based data science ecosystem. We showcased the main features of PyComplexHeatmap in rendering complex biological datasets with rich annotations. Our benchmark suggested a superior computational efficiency over the R implementation. We demonstrated its power in advanced genomics data analysis, including rendering the OncoPrint plot for cancer genomics data analysis and dissecting single-cell multiomics data of five dimensions.

AUTHOR CONTRIBUTIONS

Wubin Ding: Algorithm implementation, writing: original draft preparation and reviewing. **David Goldberg**: Data preparation. **Wanding Zhou**: writing: reviewing and editing.

ACKNOWLEDGMENTS

The study is supported by NIH grant R35GM146978.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study. The PyComplexHeatmap package and documentation are available on Github: https://github.com/DingWB/PyComplexHeatmap. Supplementary materials (figures, tables, scripts, graphical abstract, slides, videos, Chinese translated version, and updated materials) may be found in the online DOI or iMeta Science http://www.imeta.science/.

Wubin Ding[1]
David Goldberg[1]
Wanding Zhou[1,2]

[1]*Center for Computational and Genomic Medicine, The Children's Hospital of Philadelphia, Philadelphia, Pennsylvania, USA*
[2]*Department of Pathology and Laboratory Medicine, University of Pennsylvania, Philadelphia, Pennsylvania, USA*

Correspondence
Wubin Ding and Wanding Zhou, Center for Computational and Genomic Medicine, The Children's Hospital of Philadelphia, Philadelphia, PA 19104, USA.
Email: ding.wu.bin.gm@gmail.com and wanding.zhou@pennmedicine.upenn.edu

ORCID

Wubin Ding http://orcid.org/0000-0002-5355-7561

REFERENCES

1. Wolf, F. Alexander, Philipp Angerer, and Fabian J. Theis. 2018. "SCANPY: Large-Scale Single-Cell Gene Expression Data Analysis." *Genome Biology* 19: 15. https://doi.org/10.1186/s13059-017-1382-0
2. Danese, Anna, Maria L. Richter, Kridsadakorn Chaichoompu, David S. Fischer, Fabian J. Theis, and Maria Colomé-Tatché.

2021. "EpiScanpy: Integrated Single-Cell Epigenomic Analysis." *Nature Communications* 12: 5228. https://doi.org/10.1038/s41467-021-25131-3

3. Bergen, Volker, Marius Lange, Stefan Peidli, F. Alexander Wolf, and Fabian J. Theis. 2020. "Generalizing RNA Velocity to Transient Cell States Through Dynamical Modeling." *Nature Biotechnology* 38: 1408–14. https://doi.org/10.1038/s41587-020-0591-3
4. Muzellec, Boris, Maria Teleńczuk, Vincent Cabeli, and Mathieu Andreux. 2022. "PyDESeq. 2: A Python Package for Bulk RNA-Seq Differential Expression Analysis." bioRxiv. https://doi.org/10.1101/2022.12.14.520412
5. Fang, Zhuoqing, Xinyuan Liu, and Gary Peltz. 2023. "GSEApy: A Comprehensive Package for Performing Gene Set Enrichment Analysis in Python." *Bioinformatics* 39(1). https://doi.org/10.1093/bioinformatics/btac757
6. Palla, Giovanni, Hannah Spitzer, Michal Klein, David Fischer, Anna Christina Schaar, Louis Benedikt Kuemmerle, Sergei Rybakov, et al. 2022. "Squidpy: A Scalable Framework for Spatial Omics Analysis." *Nature Methods* 19: 171–78. https://doi.org/10.1038/s41592-021-01358-2
7. Angermueller, Christof, Heather J. Lee, Wolf Reik, and Oliver Stegle. 2017. "DeepCpG: Accurate Prediction of Single-Cell DNA Methylation States Using Deep Learning." *Genome Biology* 18: 67. https://doi.org/10.1186/s13059-017-1189-z
8. Gayoso, Adam, Romain Lopez, Galen Xing, Pierre Boyeau, Valeh Valiollah Pour Amiri, Justin Hong, Katherine Wu, et al. 2022. "A Python Library for Probabilistic Analysis of Single-Cell Omics Data." *Nature Biotechnology* 40: 163–66. https://doi.org/10.1038/s41587-021-01206-w
9. Gu, Zuguang, Roland, Eils, and Matthias Schlesner. 2016. "Complex Heatmaps Reveal Patterns and Correlations in Multidimensional Genomic Data." *Bioinformatics* 32: 2847–49. https://doi.org/10.1093/bioinformatics/btw313
10. Gu, Zuguang. 2022. "Complex Heatmap Visualization." *iMeta* 1: e43. https://doi.org/10.1002/imt2.43
11. Waskom, Michael. 2021. "Seaborn: Statistical Data Visualization." *Journal of Open Source Software* 6: 3021. https://doi.org/10.21105/joss.03021
12. Hunter, John D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9: 90–95.
13. Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing In Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2
14. Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming With NumPy." *Nature* 585: 357–62. https://doi.org/10.1038/s41586-020-2649-2
15. Ding, Wubin, Diljeet Kaur, Steve Horvath, and Wanding Zhou. 2023. "Comparative Epigenome Analysis Using Infinium DNA Methylation BeadChips." *Brief Bioinformatics* 24(1). https://doi.org/10.1093/bib/bbac617
16. Loyfer, Netanel, Judith Magenheim, Ayelet Peretz, Gordon Cann, Joerg Bredno, Agnes Klochendler, Ilana Fox-Fisher, et al. 2023. "A DNA Methylation Atlas of Normal Human Cell Types." *Nature* 613: 355–64. https://doi.org/10.1038/s41586-022-05580-6

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.