The tidyomics ecosystem: Enhancing omic data analyses

Hutchison WJ[1,2,+], Keyes TJ[3,4,+], *The Tidyomics Consortium*, Crowell LH[5,6], Soneson C[7,8], Yuan V[9], Nahid AA[10], Mu W[11], Park J[11] Davis ES[12], Tang M[13], Axisa PP[14], Sato N[15], Gottardo R[16,8], Morgan M[17], Lee S[18], Lawrence M[18], Hicks SC[19,20], Nolan GP[21], Davis KL[4], Papenfuss AT[1,2,*], Love M[22,11,*], Mangiola S[1,2,*]

+ These authors contributed equally, * These authors contributed equally

Abstract

The exponential growth of omic data presents challenges in data manipulation, analysis, and integration. Addressing these challenges, Bioconductor offers an extensive data analysis platform and community, while R tidy programming offers a standard data organisation and manipulation that has revolutionised data science. Bioconductor and tidy R have mostly remained independent; bridging these two ecosystems would streamline omic analysis, ease learning and encourage cross-disciplinary collaborations. Here, we introduce the *tidyomics* ecosystem—a suite of interoperable software that brings the vast tidy software ecosystem to omic data analysis.

[1] Walter and Eliza Hall Institute of Medical Research
[2] Department of Medical Biology, University of Melbourne, Parkville, VIC 3052, Australia
[3] Stanford University School of Medicine, Department of Biomedical Data Science
[4] Stanford University School of Medicine, Department of Pediatrics
[5] University of Zurich, Switzerland
[6] Centre for Genomic Regulation, Barcelona Institute of Science and Technology, Barcelona, Spain
[7] Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland
[8] SIB Swiss Institute of Bioinformatics, Basel, Switzerland
[9] Department of Statistics, The University of British Columbia
[10] Department of Biochemistry and Molecular Biology, Shahjalal University of Science and Technology, Sylhet, Bangladesh
[11] Biostatistics Department, University of North Carolina-Chapel Hill, Chapel Hill, NC, USA
[12] Bioinformatics and Computational Biology Program, University of North Carolina-Chapel Hill, Chapel Hill, NC, USA
[13] Immunitas Therapeutics
[14] Centre de Recherches en Cancerologie de Toulouse, Inserm, Toulouse, France
[15] Division of Health Medical Intelligence, Human Genome Center, The Institute of Medical Science, The University of Tokyo
[16] University of Lausanne
[17] Roswell Park Comprehensive Cancer Center
[18] Genentech, Department of Bioinformatics and Computational Biology
[19] Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
[20] Malone Center for Engineering in Healthcare, Johns Hopkins University, MD, USA
[21] Stanford University School of Medicine, Department of Pathology
[22] Genetics Department, University of North Carolina-Chapel Hill, Chapel Hill, NC, USA

Main

The advent of high-throughput genomic, transcriptomic, spatial, and multi-omic data has revolutionised biomedical research, presenting immense opportunities and formidable challenges for analysts engaged in omic data manipulation, exploration, analysis, integration, and interpretation[1]. To address these challenges, the scientific community has embraced programming languages with increasing levels of abstraction **(Figure 1A-top)**. Building upon the hardware-focused Fortran to higher-level instructions in C, scientists adopted modular data containers in languages like Python, Perl, and Java, which could represent physical entities such as genes and DNA sequences **(Figure 1A-bottom)**. Biological data analysis has gravitated toward higher-level languages oriented to data structures and statistical modelling, such as S, Matlab, R and Julia[2].

More recently, tidy R programming and the tidyverse software ecosystem[3] have transformed data science by abstracting the intricacy of data structures, and prioritising intuitive data manipulation, pipelines, and statistical models. This approach has become broadly adopted in data science educational programs, streamlining workflows and augmenting analysis transparency. Tidy data organises variables as columns and observations as rows. Tidy programming leverages a set of standardised verbs to describe data manipulation, often shared across programming languages and communities such as Python and Julia[4].

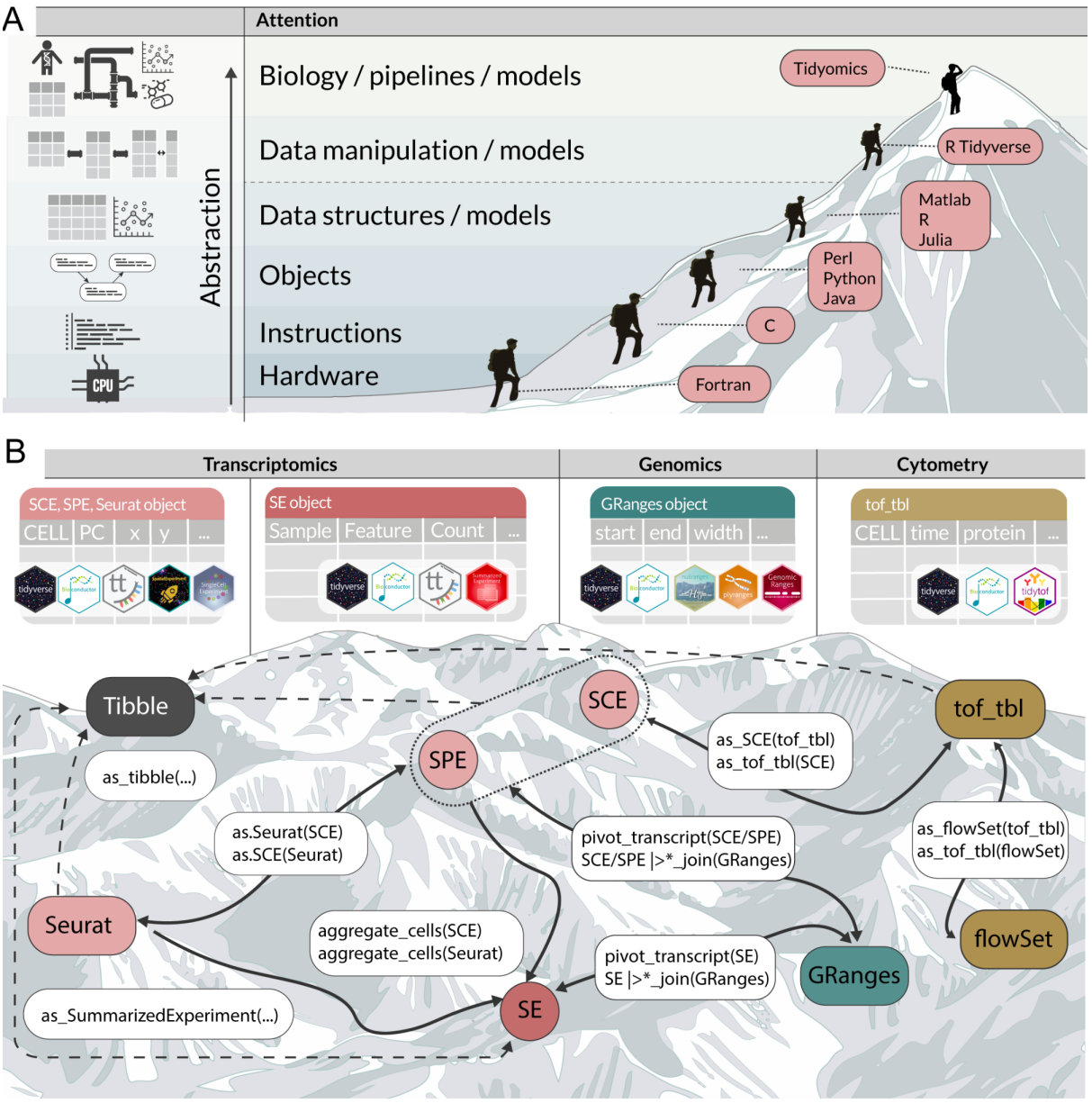

Figure 1: A: The landscape from low- to high-level languages for quantitative biology. The icons on the left represent the programmer's focus on coding languages. **B:** The landscape of interconnectivity within the tidyomics ecosystem. The top side shows the consistent data representation for the diverse data containers. The hexagonal icons represent the compatible R packages. On the bottom, the data containers are represented by rounded rectangles and functions that connect them as white boxes. SPE = SpatialExperiment, SCE = SingleCellExperiment, SE = SummarizedExperiment.

In response to the complexity of the software landscape, the Bioconductor Project[5,6] has emerged as a premier software repository for R, offering diverse tools for omic data analysis. Bioconductor is a comprehensive platform that provides international standardisation and interoperability for data processing, statistical analysis, visualisation, curated datasets, and workflows. With extensive annotation resources

and standardised data formats, Bioconductor promotes computational reproducibility and package interoperability and facilitates data interpretation through the intrinsic linkage between metadata and data.

Bioconductor has remained largely independent of the tidyverse ecosystem. Creating a bridge between these two ecosystems, by providing a tidy interface to standard data formats[7–9] and analysis[9], would shift the researcher's focus from the technical programming aspects to answering biological questions. By offering a common tidy interface for omic data manipulation, which has become a de-facto standard in data science education, the barrier to entry for analysing diverse omics data would be significantly lowered.

Here, we present *tidyomics*, an interoperable software ecosystem that bridges Bioconductor and the tidyverse. *tidyomics* is easily installable with a single homonymous meta-package. This ecosystem includes three new R packages: tidySummarizedExperiment, tidySingleCellExperiment, and tidySpatialExperiment, and five that are publicly available: plyranges[7], nullranges[10], tidyseurat[8], tidybulk[9], tidytof[11]. Importantly, this interface leaves the original data containers and methods unaltered, maintaining compatibility with existing software, maintainability and long-term support from Bioconductor. *tidyomics*, which covers genomics, transcriptomics, cytometry and spatial[12], aims to extend to proteomics[13] and multi-omics[14] to develop a technology-agnostic interface for omic data analysis.

Within Bioconductor, genomic features like accessibility and mutations are organised in ranges (e.g. genes, exons, SNPs, CpGs). Ranges are contained in GenomicRanges[15] objects, with observations as rows (e.g. gene) and variables (e.g. range width) as columns (like the BED[16] format). plyranges[7] extends dplyr verbs to GenomicRanges objects, facilitating the integration of ranges, overlap analysis, summarisation, reduction and visualisation. Plyranges interacts with nullranges[17] and bootranges[10] to perform statistical testing.

Historically, transcriptional abundance is represented as a matrix (e.g. gene-by-sample). Bioconductor provides hierarchically structured containers such as SummarizedExperiment and SingleCellExperiment[18] that link abundance with metadata. tidySummarizedExperiment and tidySingleCellExperiment packages provide a tidy data interface that organises observations (e.g. gene/sample pair) and

data (e.g. abundance, metadata) in a table, generalising the concept of the *variable*. This approach enables complex filtering, summarisation, analysis, and visualisation using the tidyverse. tidybulk[9] offers a modular analysis pipeline that fits the tidy principles for bulk and pseudobulk (from single-cell) data.

Cytometry data from mass cytometry[19], full-spectrum flow cytometry[20], and sequence-based cytometry[21] technologies are similarly organised in a cell-by-feature matrix. Bioconductor's flowCore package provides data containerisation (in flowFrame and flowSet) and manipulation[22]. Tidytof[11] interfaces flowCore with tidyverse and offers a tidy analysis framework. Tidytof is compatible with other *tidyomics* packages like tidySingleCellExperiment and tidySummarizedExperiment. Emerging proteomic data formats, like LCMS, have yet to become available on Bioconductor.

Spatial technologies such as 10X Genomics Xenium, CosMX, Mibi, and MERSCOPE have added a spatial dimension to near-, at, or sub-single-cell resolution[23]. Bioconductor's SpatialExperiment[24] is the dedicated data container for spatial cell/pixel-based transcriptomics analysis. tidySpatialExperiment offers a tidy interface to the relational data structure of SpatialExperiment. It is consistent with tidySingleCellExperiment while supporting spatial coordinates and offering specialised operations such as filtering based on geometric shapes and merging multilayer or multiomic datasets.

The tidyomic ecosystem provides a unified and interoperable framework for omic technologies, including Bioconductor (for genomics, transcriptomics and cytometry) and Seurat[8]. Through converters and integration, a network of functionalities connects various data containers within the tidyomics language **(Figure 1B)**. This harmonised approach simplifies the adoption of new technologies. It facilitates seamless container switching, decreasing the dependence on a specific framework created by domain-specific syntax and effectively increasing the umbrella of used tools.

With its unified approach, *tidyomics* brings several benefits. It enables the reuse of code for summarisation and visualisation across diverse data sources **(Figure 2A)**; this streamlines omic analyses, enhances computational reproducibility, promotes transparency, and facilitates cross-disciplinary collaborations. For example, complex

analytic tasks for genome **(Figure 2B-left)** and transcriptome **(Figure 2B-right)** data can be manipulated and visualised using modular steps assembled into a compact, legible pipeline using consistent grammar. Legibility and coding simplicity are also contributed by fewer intermediate variables and lines of code compared with standard counterparts while having no major execution-time overhead **(Figure 2C)**. Tidyomics favours functional programming (e.g., vectorisation rather than for-loops), which helps avoid bugs caused by variable updating, especially in interactive programming.

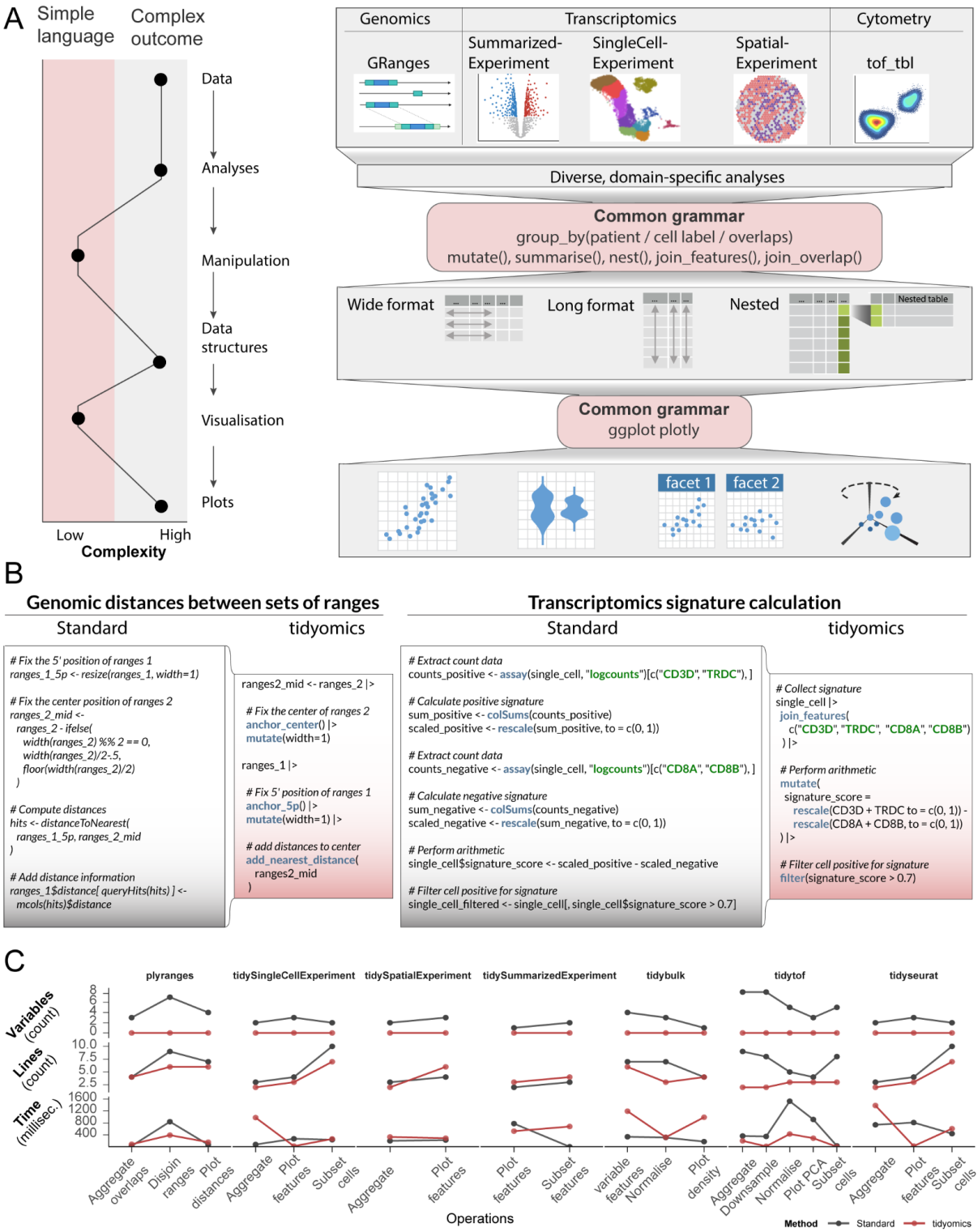

Figure 2: A: Contrast between the simplicity of the tidyomics syntax/grammar and the complex outcome and input data containers (left). Example workflows include data, biological analysis, data/results manipulation and summarisation, diverse data structures, visualisation and resulting plots (right). The pink areas include the tidyomics infrastructure that shares grammar across omics. **B:** Comparison of code readability between standard and tidyverse programming. Two tasks showcased are visualising a histogram of genomic distances (left) and calculating a multi-gene

signature from single-cell data (right). **C:** The benchmark of variables, lines of code, and time efficiency of the tidyomics ecosystem compared to standard (non-tidy) coding. The operations include common manipulations and analysis for each package (Methods).

The tidyverse and Bioconductor ecosystems are transforming data science and biological data analysis. Tidyomics bridges the gap between these ecosystems, enabling analysts to leverage the power of tidy data principles in omic analyses. This integration fosters cross-disciplinary collaborations, reduces barriers to entry for new users, and enhances code readability, reproducibility and transparency in data analysis. The tidy standard applied to biological software creates an extensible development ecosystem where independent researchers can interface new software. Ultimately, the tidyomics ecosystem, consisting of new and publicly available R packages, has the potential to significantly accelerate scientific discovery.

Code availability

The *tidyomics* meta-package is available at github.com/tidyomics/tidyomics. The tidySummarizedExperiment package is available at Bioconductor bioconductor.org/packages/tidySummarizedExperiment. The tidySingleCellExperiment package is available at Bioconductor bioconductor.org/packages/tidySingleCellExperiment. The tidySpatialExperiment package is available at github.com/william-hutchison/tidySpatialExperiment. The code used to benchmark the efficiency of standard and tidyomics workflows is available as an R vignette on github.com/tidyomics/tidyomics/blob/main/vignettes/benchmarking.Rmd

Acknowledgements

We acknowledge Bioconductor and tidyverse communities, whose software and coding paradigms this work is based on and would not be possible without. We also thank the tidyomics community for their feedback and contribution. We thank Vincent Carey for his support and feedback on the project. Also, we thank Matt Richie for his continuous support and feedback.

Author information

Authors and Affiliations

Contributions

SM and MIL conceived the study, HW and SM developed the novel tidy adapters for transcriptomics, and HW and KT performed the analyses. The tidyomics Consortium contributed to the ecosystem development and ongoing improvement. SM, MIL, ATP, DK, SCH, LM, MM, and GR acted as supervisory team. All authors contributed to the writing of the manuscript.

Corresponding author

Mangiola Stefano mangiola.s@wehi.edu.au

Michael I Love michaelisaiahlove@gmail.com

Anthony T Papenfuss papenfuss@wehi.edu.au

Ethics declarations

The authors do not have any conflict of interest.

References

1. Tarazona, S., Arzalluz-Luque, A. & Conesa, A. Undisclosed, unmet and neglected challenges in multi-omics studies. *Nature Computational Science* **1**, 395–402 (2021).
2. Roesch, E. *et al.* Julia for biologists. *Nat. Methods* **20**, 655–664 (2023).
3. Wickham, H. *et al.* Welcome to the Tidyverse. *Journal of Open Source Software* **4**, 1686 (2019).
4. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* **59**, 65–98 (2017).
5. Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5**, R80 (2004).
6. Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor.

Nat. Methods **12**, 115–121 (2015).

7. Lee, S., Cook, D. & Lawrence, M. plyranges: a grammar of genomic data transformation. *Genome Biol.* **20**, 4 (2019).
8. Mangiola, S., Doyle, M. A. & Papenfuss, A. T. Interfacing Seurat with the R tidy universe. *Bioinformatics* (2021) doi:10.1093/bioinformatics/btab404.
9. Mangiola, S., Molania, R., Dong, R., Doyle, M. A. & Papenfuss, A. T. tidybulk: an R tidy framework for modular transcriptomic data analysis. *Genome Biol.* **22**, 42 (2021).
10. Mu, W. *et al.* bootRanges: flexible generation of null sets of genomic ranges for hypothesis testing. *Bioinformatics* **39**, (2023).
11. Keyes, T. J., Koladiya, A., Lo, Y.-C., Nolan, G. P. & Davis, K. L. tidytof: a user-friendly framework for scalable and reproducible high-dimensional cytometry data analysis. *Bioinform Adv* **3**, vbad071 (2023).
12. Feng, Y. *et al.* Spatial analysis with SPIAT and spaSim to characterize and simulate tissue microenvironments. *Nat. Commun.* **14**, 2697 (2023).
13. Gatto, L. & Christoforou, A. Using R and Bioconductor for proteomics data analysis. *Biochimica et Biophysica Acta (BBA)-Proteins and* (2014).
14. Odom, G. J., Colaprico, A., Silva, T. C., Chen, X. S. & Wang, L. PathwayMultiomics: An R Package for Efficient Integrative Analysis of Multi-Omics Datasets With Matched or Un-matched Samples. *Front. Genet.* **12**, 783713 (2021).
15. Lawrence, M. *et al.* Software for computing and annotating genomic ranges. *PLoS Comput. Biol.* **9**, e1003118 (2013).
16. Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841–842 (2010).
17. Davis, E. S. *et al.* matchRanges: generating null hypothesis genomic ranges via covariate-matched sampling. *Bioinformatics* **39**, (2023).
18. Amezquita, R. A. *et al.* Orchestrating single-cell analysis with Bioconductor. *Nat. Methods* **17**, 137–145 (2020).
19. Bendall, S. C. *et al.* Single-cell mass cytometry of differential immune and drug

responses across a human hematopoietic continuum. *Science* **332**, 687–696 (2011).

20. Jaimes, M. C. *et al.* Full spectrum flow cytometry and mass cytometry: A 32-marker panel comparison. *Cytometry A* **101**, 942–959 (2022).
21. Salomon, R., Martelotto, L., Valdes-Mora, F. & Gallego-Ortega, D. Genomic Cytometry and new modalities for deep single-cell interrogation. *Cytometry A* **97**, 1007–1016 (2020).
22. Ko, M. E. *et al.* FLOW-MAP: a graph-based, force-directed layout algorithm for trajectory mapping in single-cell time course datasets. *Nat. Protoc.* **15**, 398–420 (2020).
23. Wang, Y. *et al.* Spatial transcriptomics: Technologies, applications and experimental considerations. *Genomics* 110671 (2023).
24. Righelli, D. *et al.* SpatialExperiment: infrastructure for spatially-resolved transcriptomics data in R using Bioconductor. *Bioinformatics* **38**, 3128–3131 (2022).

Methods

tidySummarizedExperiment

The tidySummarizedExperiment method introduces an innovative and versatile approach for representing and managing bulk data, offering an alternative to the conventional methodologies commonly employed in SummarizedExperiment. This novel approach incorporates adaptors tailored to popular data manipulation and visualisation packages, including dplyr, tidyr, ggplot, and plotly. Crucially, the core structure of the SummarizedExperiment object remains unaltered, ensuring seamless compatibility with existing analytical workflows.

Data is represented as a long table, wherein observations are defined by the abundance of a feature/sample pair, while variables encompass feature and sample-related metadata. The fundamental columns comprising this representation consist of the feature and sample columns. Importantly, when any of these essential columns are absent from the output of a given operation (e.g., select) or when a summarised version of these columns is generated, an independent table is returned. This supplementary table adheres to the structure of the SummarizedExperiment tidy representation, facilitating separate analysis or visualisation as needed.

Furthermore, in instances where the subset of observations does not correspond to a rectangular slice of the feature/sample matrix, a table is provided to accommodate such scenarios.

With the tidySummarizedExperiment approach, newly created or joined columns, such as those obtained from a metadata table, are automatically incorporated into the colData, rowData, or assays based on their alignment with feature or sample identifiers. This versatile mechanism extends the "variable" concept, enabling a manipulation, displaying, or visualisation of sample, feature, and abundance information with consistent grammar. Notably, the columns for the sample and feature identifiers and genomic ranges are designated as read-only to preserve data integrity.

tidySingleCellExperiment

The tidySingleCellExperiment method presents a novel approach for representing and manipulating single-cell data, providing an alternative to the conventional methods commonly used in SingleCellExperiment. The main goal and property of the API are consistent with tidySummarizedExperiment.

On the contrary, the tidy representation is limited to cell-wise information, organised into an easily interpretable table format. Cell information includes metadata and reduced dimensions (e.g., principal components and UMAP dimensions). To access transcript-level information, users can utilise the `join_features` function. This function enriches the data by incorporating transcript identifiers, transcript abundance, and transcript-wise annotations, including gene length, genomic coordinates, and functional annotations, as additional columns of the cell metadata.

The emphasis on cell-wise information over transcript-wise information is driven by the priority to facilitate ease of use, data summarisation, information integration, and data visualisation in the context of cell analysis. By focusing on cell-wise information, the abstraction avoids unnecessary complexity arising from including feature-level information (e.g., genes, proteins, genomic regions) by default, especially when performing cell-wise information subsetting.

The tibble abstraction employed in tidySingleCellExperiment consists of two types of columns: editable columns, which allow for user interaction and modification, and

view-only columns, which encompass data-derived variables, such as reduced dimensions. Integrating all cell-wise information, including reduced dimensions, within a single tibble representation enables seamless data visualisation, filtering, and manipulation. Importantly, this design ensures compatibility with the tidyverse ecosystem, enabling data manipulation and plotting using routines from dplyr, tidyr, ggplot2, and plotly. Furthermore, adopting this abstraction allows users to operate on the data as if it were a standard tibble while preserving compatibility with any other algorithms or tools that utilise the SingleCellExperiment framework. This approach ensures full backward compatibility and facilitates seamless integration into existing workflows.

tidySingleCellExperiment shares the same grammar and data representation as tidyseurat, allowing users to use tidy code with SingleCellExperiment and Seurat data containers.

It is important to highlight that operations that appear to edit the tidySingleCellExperiment object, resembling a tabular (e.g., tibble) object, actually update the underlying SingleCellExperiment object, thus maintaining data integrity and consistency.

tidySpatialExperiment

tidySpatialExperiment provides a tidy R abstraction (tibble) of SpatialExperiment objects. Similarly to tidySingleCellExperiment, it provides cell-wise information, including cell metadata, spatial coordinates and metadata, and reduced dimensions. All information can be processed with tools provided by dplyr, tidyr, ggplot2 and plotly. tidySpatialExperiment provides the `join_features` function to append the specified features to the cell-wise information and, consequently, the tibble abstraction. The `aggregate_cells` function is provided to combine cells by shared variables and aggregate feature counts.

Benchmarking

Benchmarking was performed to emulate common data analysis tasks in standard and tidyomics workflows. Briefly, the aggregate-cells-by-sample operation aggregates the feature counts of cells within each sample. The plot features per cell operation plots the distribution of summed feature counts for each cell within each

sample. The subset cells by feature operation subsets cells by feature signature threshold. The subset-features-by-mean-count operation subsets features by mean count threshold. The plot-features-per-sample operation plots the distribution of feature counts for each sample. The normalise-features operation normalises feature counts across samples. The plot-normalised-feature-density operation plots the density of normalised features for each sample. The identify-variable-features operation identifies the most variable features for each cell type. The aggregate-overlaps operation identifies and aggregates overlapping regions. The plot-feature-set-distances operation calculates and plots the distance between feature sets. The group-disjoin-ranges operation finds disjoint regions within groups of features and subsets overlaps. The downsample-cells operation randomly subsets cells from each sample. The plot-PCA operation calculates and plots principal components. The benchmarking operations were run using R v4.3.1, plyranges v1.20.0, tidySingleCellExperiment v1.11.3, tidySpatialExperiment v0.0.3, tidySummarizedExperiment v1.10.0, tidybulk v1.12.0, tidytof v0.0.0, and tidyseurat v0.6.1.

Each benchmarking operation was executed 100 times using the microbenchmark package, and the mean time elapsed was recorded. The variable count was calculated as the number of times a new variable was created to store data. The line count was calculated as the number of lines created following indentation best practice and as required to maintain a line width below 101 characters.