Multi-task benchmarking of single-cell multimodal omics integration methods

Chunlei Liu[1,3], Sichang Ding[1,5], Hani Jieun Kim[1,3], Siqu Long[1,3], Di Xiao[1,3], Shila Ghazanfar[2,3,4] and Pengyi Yang[1,2,3,4]*

[1] Computational Systems Biology Unit, Children's Medical Research Institute, Faculty of Medicine and Health, University of Sydney, Westmead, NSW 2145, Australia
[2] School of Mathematics and Statistics, Faculty of Science, University of Sydney, Sydney, NSW 2006, Australia
[3] Sydney Precision Data Science Centre, University of Sydney, Sydney, NSW 2006, Australia
[4] Charles Perkins Centre, University of Sydney, Sydney, NSW 2006, Australia
[5] School of Computer Science, Faculty of Engineering, University of Sydney, Sydney, NSW 2006, Australia

* Correspondence (pengyi.yang@sydney.edu.au)

Abstract
Single-cell multimodal omics technologies have empowered the profiling of complex biological systems at a resolution and scale that were previously unattainable. These biotechnologies have propelled the fast-paced innovation and development of data integration methods, leading to a critical need for their systematic categorisation, evaluation, and benchmark. Navigating and selecting the most pertinent integration approach poses a significant challenge, contingent upon the tasks relevant to the study goals and the combination of modalities and batches present in the data at hand. Understanding how well each method performs multiple tasks, including dimension reduction, batch correction, cell type classification and clustering, imputation, feature selection, and spatial registration, and at which combinations will help guide this decision. This study aims to develop a much-needed guideline on choosing the most appropriate method for single-cell multimodal omics data analysis through a systematic categorisation and comprehensive benchmarking of current methods.

Key words: single-cell multimodal omics, multi-task, integration methods, benchmarking

Introduction

The recent development of single-cell multimodal omics technologies has revolutionised our ability to simultaneously profile multi-layered molecular programs at a global scale in individual cells[1,2]. The availability of single-cell multimodal omics data provides new opportunities to investigate molecular programs that underlie cell identity, cell-fate decisions, and disease mechanisms at a resolution previously inaccessible from studying bulk cell populations[1–3]. A considerable number of technologies and platforms have been developed, which have led to the generation of a variety of combinations of data modalities each capturing a unique molecular feature in the cell. To date, prevalent modalities encompass gene expression (typically denoted as RNA), surface protein abundance (through antibody derived tags [ADT]), and chromatin accessibility (through assay for transposase-accessible chromatin [ATAC]), and can be profiled jointly by popular single-cell multimodal omics technologies such as CITE-seq[4], SHARE-seq[5], and TEA-seq[6] to name a few (**Fig. 1a**).

Integrating modalities of data generated from single-cell multimodal omics technologie is essential and greatly impacts the utility of such data for downstream biological interpretation[7–9]. A fast-growing number of computational methods have been developed to integrate various combinations of data modalities captured by different single-cell multimodal omics technologies. Nevertheless, there is a lack of study that comprehensively outlines the similarities and differences between current methods including the data integration strategies, the tasks they intend to address, and the data modalities they can integrate. Furthermore, while a few works have been conducted to compare integration methods for cancer studies using both bulk and single-cell omics data[10,11], there is a lack of systematic evaluation and benchmark of the currently available methods specifically designed for single-cell multimodal omics data integration for their performance on the broad range of tasks and data modalities they are designed to handle. These, together, have created a significant challenge for navigating and selecting the most pertinent integration method for single-cell multimodal omics data analysis.

To address the challenge of selecting the most appropriate methods for single-cell multimodal omics data analysis, here we set out to systematically categorise and comprehensively benchmark the current integration methods on common computational tasks and data modalities. Building on previous works[7,12] and based on input data structure and modality combination, we define four prototypical single-cell multimodal omics data integration categories, including “vertical”, “diagonal”, “mosaic”, and “cross” integration (**Fig. 1b**). Depending on the applications, we further introduce seven common tasks that many methods are designed to address (**Fig. 1c**), including (1) dimension reduction, (2) batch correction, (3) clustering, (4) classification, (5) feature selection, (6) imputation, and (7) spatial registration. Using panels of evaluation metrics each tailor-made for a specific task (**Fig. 1d**), we evaluated 39 integration methods (**Fig. 1e**) across the four data integration categories on 64 real datasets and 22 simulated datasets. In particular, we include 18 vertical integration methods, 13 diagonal integration tools, 12 mosaic integration tools, and 15 cross integration tools. This registered report will provide a comprehensive view of methods designed for single-cell multimodal omics data integration, and serve as (i) a much-needed guide for single-cell integration across modalities to the research community and (ii) a foundation to foster future research and application of computational methods for single-cell multimodal omics data analysis.

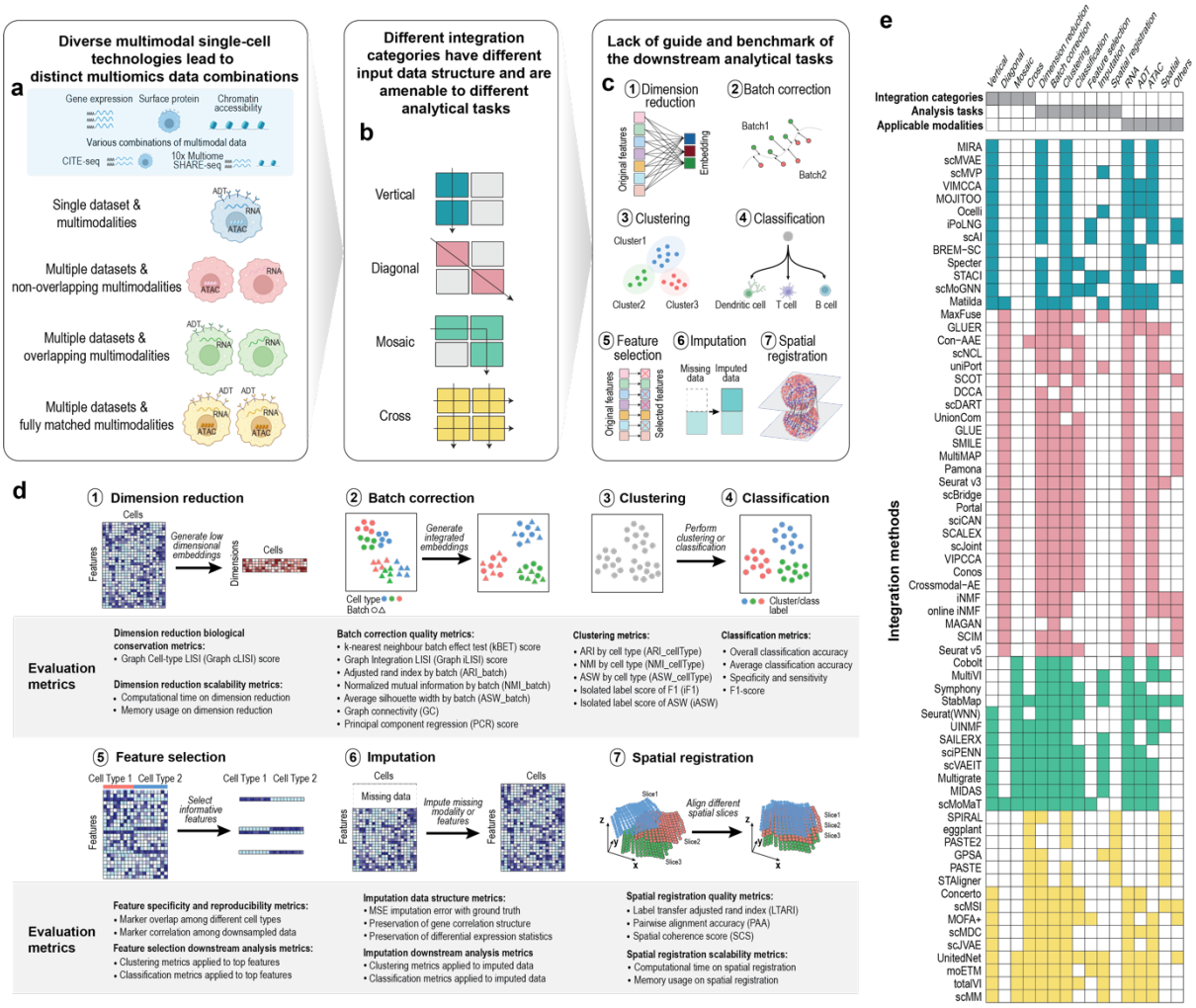

Fig. 1: An overview of benchmarking for single-cell multimodal omics integration methodologies. a, diverse technological combinations arising from different modalities. **b**, four prototypical single-cell multimodal omics data integration categories. **c**, schematics of common tasks for single-cell multimodal omics analysis. **d**, schematics for each task and the panel of evaluation metrics tailored for its performance evaluation. **e**, summary of integration methods and their alignment with specific integration categories, computational tasks, and modality capacities.

Detailed experimental design plan

Summary of integration methods in each category and task

The evaluation of integration methods from the four integration categories and seven tasks is summarised in **Fig. 2a-d**. The following criteria were used for selecting integration methods for benchmarking:

- ***Reasonable speed and memory requirements.*** Methods characterised by excessively slow speeds or high memory consumption will be excluded.
- ***High reproducibility.*** Only methods demonstrating consistent results when applied to varied datasets, ensuring robustness and reliability, were included in the benchmarking process.
- ***Comprehensive documentation and tutorials.*** We will exclude methods lacking detailed descriptions of input data and/or running instructions.
- ***Active maintenance.*** Preference will be given to methods that are maintained and updated.

- ***Joint latent space or graph generation.*** We will exclude methods that do not generate a joint latent space or graph except those for spatial registration, as this is an overwhelmingly common capability among most methods.
- ***Broad applicability.*** Methods that are designed exclusively for specific types of data, such as CyTOF, and are not broadly applicable will not be included.
- ***Platform accessibility.*** We will focus on methods compatible with Python or R.
- ***Ease of installation.*** Methods with special computing requirements and are complex to install will be disregarded.

The above selection criteria led to triage of 39 integration methods. The details of the 64 real datasets and 22 simulated datasets that will be used for the benchmarking are included in the "Datasets and preprocessing" section, and the experimental settings for each of the 39 integration methods are described in the "Single-cell multimodal omics data integration methods" section.

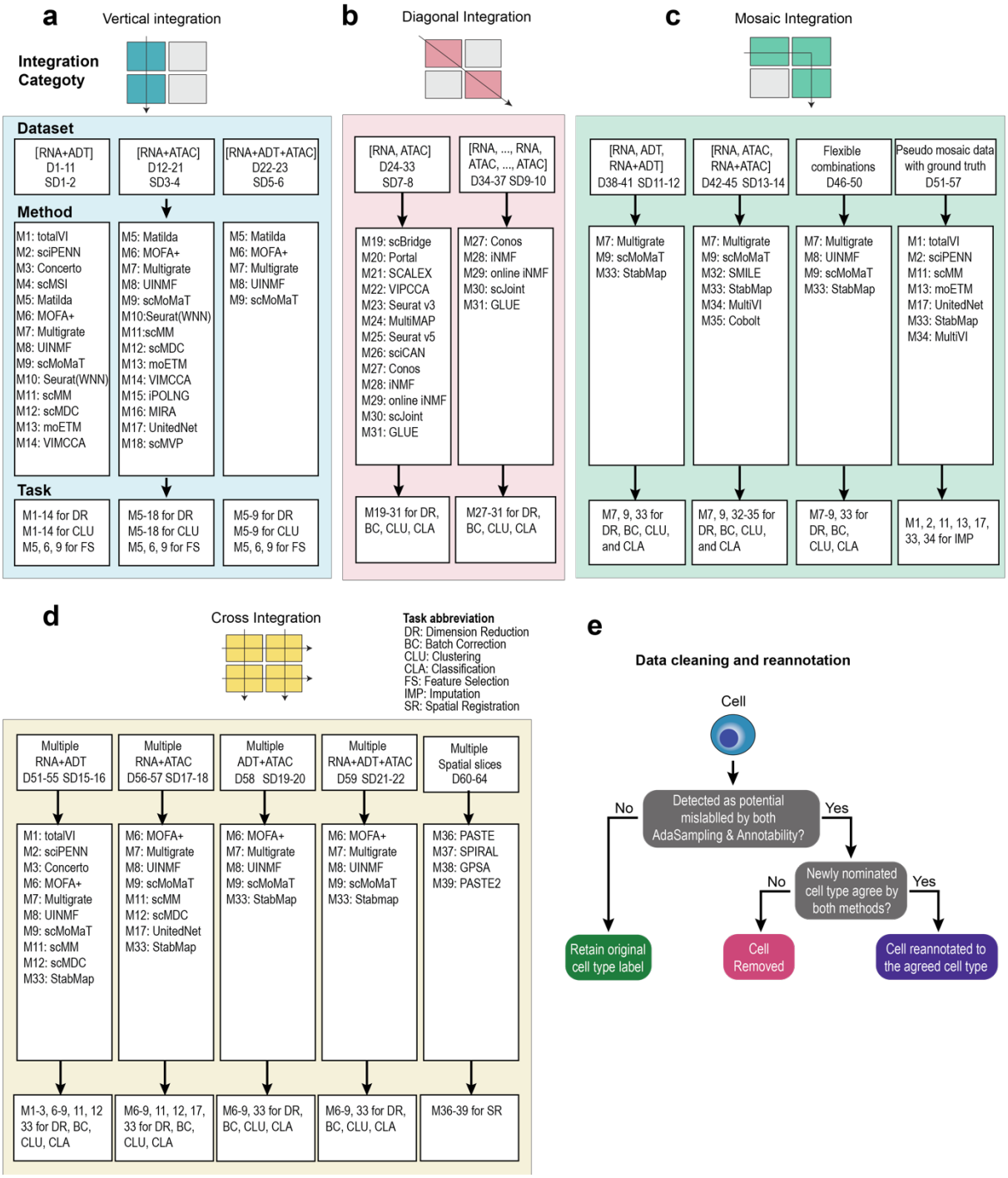

Fig. 2: Summary of the 39 integration methods included in the benchmark evaluation. Methods are grouped by their integration category including **a**, vertical integration, **b**, diagonal integration, **c**, mosaic integration, and **d**, cross integration. Datasets and tasks they are evaluated on are indexed. **e**, data cleaning and reannotation procedure for quality control cell type annotation in each dataset.

Datasets and preprocessing

Experimental datasets

We have created a total of 64 datasets from 21 data sources (see Section "Data availability") generated using a variety of single-cell multimodal omics technologies. Details of each dataset grouped by integration categories are provided below.

Datasets for vertical integration

D1: consists of samples generated by the "Cambridge Medical Centre" from data source 1. It contains paired RNA and ADT data modalities of 30,313 cells from 17 cell types with 24,737 genes and 192 proteins.

D2: consists of samples generated by the "NCL Medical Centre" from data source 1. It contains paired RNA and ADT data modalities of 64,262 cells from 18 cell types, with 24,737 genes and 192 proteins.

D3: consists of "HD1_Adult" samples from data source 2. It contains paired RNA and ADT data modalities of 8,650 cells from 30 cell types with 33,538 genes and 189 proteins.

D4: consists of "HD2_Adult" samples from data source 2. It contains paired RNA and ADT data modalities of 9,538 cells from 29 cell types with 33,538 genes and 189 proteins.

D5: consists of "HD3_Adult" samples from data source 2. It contains paired RNA and ADT data modalities of 10,423 cells from 30 cell types with 33,538 genes and 189 proteins.

D6: consists of "Batch1" samples from data source 3. It contains paired RNA and ADT data modalities of 67,090 cells from 31 cell types with 33,538 genes and 228 proteins.

D7: consists of "Batch2" samples from data source 3. It contains paired RNA and ADT data modalities of 94,674 cells from 31 cell types with 33,538 genes and 228 proteins.

D8: consists of "batch1" samples from data source 5. It contains paired RNA and ADT data modalities of 14,468 cells from 27 cell types with 17,009 genes and 25 proteins.

D9: consists of "batch2" samples from data source 5. It contains paired RNA and ADT data modalities of 16,204 cells from 27 cell types with 17,009 genes and 25 proteins.

D10: consists of "PBMC" samples from data source 6. It contains paired RNA and ADT data modalities of 7,108 cells from 11 cell types with 33,514 genes and 52 proteins.

D11: consists of "Lung" samples from data source 6. It contains paired RNA and ADT data modalities of 3,362 cells from 14 cell types with 33,514 genes and 52 proteins.

D12: consists of all samples from data source 7. It contains paired RNA and ATAC data modalities of 34,774 cells from 23 cell types with 23,296 genes and 344,592 ATAC peaks.

D13: consists of "AdBrainCortex" samples from data source 8. It contains paired RNA and ATAC data modalities of 10,309 cells from 23 cell types with 33,160 genes and 244,544 ATAC peaks.

D14: consists of "P0_BrainCortex" samples from data source 8. It contains paired RNA and ATAC data modalities of 5,081 cells from 19 cell types with 19,322 genes and 229,429 ATAC peaks.

D15: consists of all samples from data source 11. It contains paired RNA and ATAC data modalities of 11,787 cells from 13 cell types with 36,601 genes and 108,344 ATAC peaks.

D16: consists of “Adol1” samples from data source 12. It contains paired RNA, ATAC data modalities of 3,188 cells from 14 cell types with 30,201 genes, 304,034 ATAC peaks.

D17: consists of “Adult1” samples from data source 12. It contains paired RNA, ATAC data modalities of 1,541 cells from 11 cell types with 30,201 genes, 304,034 ATAC peaks.

D18: consists of “Child1” samples from data source 12. It contains paired RNA, ATAC data modalities of 5,418 cells from 13 cell types with 30,201 genes, 304,034 ATAC peaks.

D19: consists of “EaFet1” samples from data source 12. It contains paired RNA, ATAC data modalities of 2,801 cells from 12 cell types with 30,201 genes, 304,034 ATAC peaks.

D20: consists of “Inf1” samples from data source 12. It contains paired RNA, ATAC data modalities of 2,723 cells from 14 cell types with 30,201 genes, 304,034 ATAC peaks.

D21: consists of “LaFet1” samples from data source 12. It contains paired RNA, ATAC data modalities of 3,102 cells from 12 cell types with 30,201 genes, 304,034 ATAC peaks.

D22: consists of “Control” samples from data source 14. It contains paired RNA, ADT, and ATAC data modalities of 7,624 cells from 26 cell types with 36,495 genes, 210 proteins, and 68,963 ATAC peaks.

D23: consists of “Stim” samples from data source 14. It contains paired RNA, ADT, and ATAC data modalities of 6,139 cells from 27 cell types with 36,495 genes, 210 proteins, and 68,963 ATAC peaks.

Datasets for diagonal integration

D24: consists of unpaired RNA and ATAC data modalities from data source 7. Both modalities contain 34,774 cells from 23 cell types. The RNA data modality contains 23,296 genes and the ATAC data modality contains 344,592 peaks.

D25: consists of unpaired RNA and ATAC data modalities of ‘AdBrainCortex’ samples from data source 8. Both modalities contain 10,309 cells from 23 cell types. The RNA data modality contains 33,160 genes and the ATAC data modality contains 244,544 peaks.

D26: consists of unpaired RNA and ATAC data modalities of ‘P0_BrainCortex’ samples from data source 8. Both modalities contain 5,081 cells from 19 cell types. The RNA data modality contains 19,322 genes and the ATAC data modality contains 229,429 peaks.

D27: consists of unpaired RNA and ATAC data modalities from data source 11. Both modalities contain 11,787 cells from 13 cell types. The RNA data modality contains 36,601 genes and the ATAC data modality contains 108,344 peaks.

D28: consists of unpaired RNA and ATAC data modalities from data source 14. The RNA data modality of “Control” samples contains 7,624 cells from 26 cell types with 36,495 genes and the ATAC data modality contains 6,139 cells from 27 cell types with 68,963 peaks.

D29: consists of unpaired RNA and ATAC data modalities from data source 15. The RNA data modality contains 6,314 cells from 13 cell types with 36,601 genes and the ATAC data modality contains 6,548 cells from 12 cell types with 94,986 peaks.

D30: consists of unpaired RNA and ATAC data modalities from data source 11 and 14. The RNA data modality extracted from data source 11 contains 11,787 cells from 13 cell types with 36,601 genes. The ATAC data modality extracted from the 'Control' samples of data source 14 contains 6,139 cells from 26 cell types with 68,963 peaks.

D31: consists of unpaired RNA and ATAC data modalities from two different sources including "Control" samples of source 14 and "batch1" samples of source 15. The RNA data modality contains 7,624 cells from 26 cell types with 36,495 genes and ATAC data modality contains 6,314 cells from 13 cell types with 99,658 peaks.

D32: consists of unpaired RNA and ATAC data modalities of "E13" samples from data source 21. The RNA data modality contains 2,187 cells from 14 cell types with 17,058 genes and the ATAC data modality contains 2,187 cells from 8 cell types with 32,437 peaks.

D33: consists of unpaired RNA and ATAC data modalities of "P22" brain from data source 21. The RNA data modality contains 9,215 cells from 11 cell types with 22,914 genes and the ATAC data modality contains 9,215 cells from 14 cell types with 121,068 peaks.

D34: consists of 13 batches of unpaired RNA and ATAC data modalities from data source 9. Both data modalities contain identical numbers of cells (i.e., 6,224, 6,740, 4,279, 4,220, 6,111, 4,895, 6,781, 4,325, 1,679, 1,771, 8,023, 9,876, and 4,325) and numbers of cell types (i.e., 21, 21, 19, 20, 17, 16, 21, 20, 18, 18, 19, 19, and 18) across the 13 batches. The RNA data modality contains 13,431 genes and the ATAC data modality contains 11,6490 peaks.

D35: consists of 9 batches of unpaired RNA and ATAC data modalities from data source 10. Both data modalities contain identical numbers of cells (i.e., 8,343, 10,968, 3,977, 6,301, 5,467, 9,584, 7,360, 3,822, and 5,204) and numbers of cell types (i.e., 32, 34, 24, 31, 31, 31, 34, 26, and 32) across the 9 batches. The RNA data modality contains 32,285 genes and the ATAC data modalities contain 162,669, 156,487, 172,779, 169,478, 159,345, 185,545, 186494, 157,105, 170,504 peaks, respectively, and 263,550 consolidated peaks.

D36: consists of 2 batches of unpaired RNA and ATAC data modalities from data source 14. Batch 1 and 3 from "Control" samples both contain 7,624 cells from 26 cell types, and batch 2 and 4 from "Stim" samples both contain 6,139 cells from 27 cell types. The RNA data modality contains 36,495 genes and the ATAC data modality contains 68,963 peaks.

D37: consists of 3 batches of unpaired RNA and ATAC data modalities from data source 15. Both data modalities contain identical numbers of cells (i.e., 6,314, 6,548, and 6,547) and numbers of cell types of 13,12,13 across the 3 batches. The RNA data modalities contain 36,601, 36,601, 36,601 genes and the ATAC data modalities contain 99,658, 94,986, 83,234 peaks, respectively, and 109,087 consolidated peaks.

Datasets for mosaic integration

D38: sourced from data source 1, consists of 3 data batches: one with only RNA modality (24,737 genes), a second with both RNA (24,737 genes) and ADT (192 proteins) modalities, and a third with only ADT modality (192 proteins). It contains 30,313, 64,262, and 30,313 cells from 17, 18, and 17 cell types, respectively.

D39: sourced from data source 2, consists of 3 data batches: one with only RNA modality (33,538 genes), a second with both RNA (33,538 genes) and ADT (189 proteins) modalities, and a third with only ADT modality (189 proteins). It contains 8,650, 9,538, and 10,423 cells from 30, 29, and 30 cell types, respectively.

D40: sourced from data source 3, consists of 3 data batches: one with only RNA modality (33,538 genes), a second with both RNA (33,538 genes) and ADT (228 proteins) modalities, and a third with only ADT modality (228 proteins). It contains 67,090, 94,674, and 67,090 cells from 31, 31, and 31 cell types, respectively.

D41: sourced from data source 5, consists of 3 data batches: one with only RNA modality (17,009 genes), a second with both RNA (9,731 genes) and ADT (25 proteins) modalities, and a third with only ADT modality (25 proteins). It contains 14,468, 16,204, and 14,468 cells from 27, 27, and 27 cell types, respectively.

D42: sourced from data source 7, consists of 3 data batches: one with only RNA modality (23,296 genes), a second with both RNA (23,296 genes) and ATAC (344,592 peaks) modalities, and a third with only ATAC modality (344,592 peaks). It contains 34,774 cells from 23 cell types for all 3 batches.

D43: sourced from data source 8, consists of 3 data batches: one with only RNA modality (33,160 genes), a second with both RNA (33,160 genes) and ATAC (244,544 peaks) modalities, and a third with only ATAC modality (244,544 peaks). It contains 10,309 cells from 23 cell types for all 3 batches.

D44: sourced from data source 8, consists of 3 data batches: one with only RNA modality (19,322 genes), a second with both RNA (19,322 genes) and ATAC (229,429 peaks) modalities, and a third with only ATAC modality (229,429 peaks). It contains 5,081 cells from 19 cell types for all 3 batches.

D45: sourced from data source 11, consists of 3 data batches: one with only RNA modality (36,601 genes), a second with both RNA (36,601 genes) and ATAC (108,344 peaks) modalities, and a third with only ATAC modality (108,344 peaks). It contains 11,787 cells from 13 cell types for all 3 batches.

D46: sourced from data sources 4 and 9, consists of 3 data batches: one with paired RNA (13,953 genes) and ADT (134 proteins) modalities, a second with paired RNA (13,431 genes) and ATAC (116,490 peaks) modalities, and a third with only RNA modality (13,953 genes). It contains 10,465, 9,876, and 5,227 cells from 38, 19, and 37 cell types, respectively.

D47: sourced from data sources 4 and 9, consists of 3 data batches: one with paired RNA (13,953 genes) and ADT (134 proteins) modalities, a second with paired RNA (13,431 genes) and ATAC (116,490 peaks) modalities, and a third with only RNA modality (13,953 genes). It contains 11,473, 8,023, and 11,035 cells from 34, 19, and 36 cell types, respectively.

D48: sourced from data sources 4 and 9, consists of 3 data batches: one with paired RNA (13,953 genes) and ADT (134 proteins) modalities, a second with paired RNA (13,431 genes) and ATAC (116,490 peaks) modalities, and a third with only ATAC modality (116,490 peaks). It contains 9,521, 4,895, and 9,876 cells from 37, 16, and 19 cell types, respectively.

D49: sourced from data sources 4 and 9, consists of 3 data batches: one with paired RNA (13,953 genes) and ADT (134 proteins) modalities, a second with paired RNA (13,431 genes) and ATAC (116,490 peaks) modalities, and a third with only ADT modality (134 proteins). It contains 4,978, 4,325, and 5,227 cells from 41, 20, and 37 cell types, respectively.

D50: sourced from data sources 4 and 9, consists of 5 data batches: one with paired RNA (13,953 genes) and ADT (134 proteins) modality, a second with paired RNA (13,431 genes) and ATAC (116,490 peaks) modality, a third with only RNA modality (13,953 genes), a fourth with only ADT modality (134 proteins), and a fifth with only ATAC modality (116,490 peaks). It contains 10,465, 9,876, 11,473, 11,035, and 8,023 cells from 38, 19, 34, 36, and 19 cell types, respectively.

Datasets for cross integration

D51: consists of 2 batches of CITE-seq data generated from data source 1. Batch 1 consists of samples from the "Cambridge Medical Centre" and contains 30,313 cells from 17 cell types and batch 2 consists of samples from the "NCL Medical Centre" and contains 64,262 cells from 18 cell types. Both batches consist of 24,737 genes and 192 proteins.

D52: consists of 3 batches of CITE-seq data generated from data source 2. Batch 1 consists of samples from "HD1_Adult" and contains 8,650 cells from 30 cell types; batch 2 consists of samples from "HD2_Adult" and contains 9,538 cells from 29 cell types; and batch 3 consists of samples from "HD3_Adult" and contains 10,423 cells from 30 cell types. All batches consist of 33,538 genes and 189 proteins.

D53: consists of 2 batches of CITE-seq data generated from data source 3. Batch 1 consists of samples from "batch1" and contains 67,090 cells from 31 cell types and batch 2 consists of samples from "batch2" and contains 94,674 cells from 31 cell types. Both batches consist of 33,538 genes and 228 proteins.

D54: consists of 12 batches of CITE-seq data generated from data source 4. The 12 batches contain 5,227, 4,978, 6,106, 10,465, 5,584, 9,122, 9,521, 11,035, 11,473, 5,456, 3,929, and 7,365 cells, which correspond to 37, 41, 36, 38, 30, 34, 37, 36, 34, 39, 33, and 36 cell types, respectively. All batches share the same set of features including 13,953 genes and 134 proteins.

D55: consists of 2 batches of CITE-seq data generated from data source 5. Batch 1 consists of samples from "batch1" and contains 14,468 cells from 27 cell types, and batch 2 consists of samples from "batch2" and contains 16,204 cells from 27 cell types. Both batches share the same set of features including 17,009 genes and 25 proteins.

D56: consists of 13 batches of 10x multiome data generated from data source 9. The 13 batches contain 6,224, 6,740, 4,279, 4,220, 6,111, 4,895, 6,781, 4,325, 1,679, 1,771, 8,023, 9,876, and 4,325 cells, which correspond to 21, 21, 19, 20, 17, 16, 21, 20, 18, 18, 19, 19, and 18 cell types, respectively. All batches share the same set of features including 13,431 genes and 11,6490 ATAC peaks.

D57: consists of 9 batches of 10x multiome data generated from data source 10. The 9 batches contain 8,343, 10,968, 3,977, 6,301, 5,467, 9,584, 7,360, 3,822, and 5,204 cells, which correspond to 32, 34, 24, 31, 31, 31, 34, 26, and 32 cell types, respectively. All batches share the same set of RNA features with 32,285 genes while the ATAC data modalities contain 162,669, 156,487, 172,779, 169,478, 159,345, 185,545, 186,494, 157,105, 170,504 peaks (263,550 consolidated peaks), respectively.

D58: consists of 2 batches of ASAP-seq data generated from data source 13. Batch 1 contains 3,517 cells from 4 cell types, and batch 2 contains 4,849 cells from 4 cell types. ATAC data modalities contain 21,994, 26,808 peaks (21,344 common peaks), while ADT data modalities contain 227, 227 proteins.

D59: consists of 2 batches of DOGMA-seq data generated from data source 14. Batch 1 contains 7,624 cells from 26 cell types, and batch 2 contains 6,139 cells from 27 cell types. Both batches share the same set of features including 36,495 genes, 210 proteins, and 68,963 ATAC peaks.

Datasets for spatial integration

D60: consists of 12 batches of spatial transcriptomics data extracted from mouse squamous cell carcinoma from data source 16. Each batch of data contains RNA expression and spatial coordinates, processed using the Visium platform. The numbers of cells in each batch are 12,453, 12,675, 12,830, 8946, 9759, 8287, 10,319, 10,176, 8,399, 7,659, 9,949, and 8,359, and the corresponding numbers of genes are 666, 645, 638, 584, 517, 517, 1,125, 1,035, 828, 545, 619, and 460. The number of spot labels in each batch are 12, 12, 12, 8, 8, 8, 12, 12, 12, 8, 8, 7.

D61: includes 12 batches of spatial transcriptomics data extracted from the human dorsolateral prefrontal cortex from data source 17. Each batch of data contains RNA expression and spatial coordinates, processed using the Visium platform. The number of cells in each batch are 9,948, 9,256, 10,486, 10,104, 10,461, 9,972, 10,996, 10,609, 11,381, 12,381, 10,629, and 10,778, and the corresponding numbers of genes are 4,220, 4,379, 4,786, 4,593, 3,635, 3,483, 4,092, 3,888, 3,611, 3,635, 3,566, and 3,431. The number of spot labels in each batch are 7, 7, 7, 7, 5, 5, 5, 5, 7, 7, 7, 7.

D62: includes 2 batches of spatial transcriptomics data of human breast cancer extracted from data source 18. Each batch of data contains RNA expression and spatial coordinates, processed using Xenium platform. The number of cells in each batch are 167,780, 118,752, and the 2 batches both have 541 genes and 20 spot labels.

D63: includes 4 batches of spatial transcriptomics data of drosophila embryos extracted from data source 19. Each batch of data contains RNA expression and spatial coordinates, processed using Stereo-seq. The number of cells in each batch are 925, 1,272, 1,263, 1,248, and the corresponding numbers of genes and cell types are consistently 13,668 and 10, respectively.

D64: includes 8 batches of spatial transcriptomics data of mouse brain extracted from data source 20. Each batch of data contains RNA expression and spatial coordinates, processed using MERFISH. The number of cells in each batch are 9,571, 11,047, 10,890, 9,665, 9,617, 8,823, 15,694, 24,790 and the corresponding numbers of genes and cell types are consistently 374 and 13, respectively.

All datasets are listed in **Supplementary Table 1**, and the datasets employed for each method and task are summarised in **Fig. 2a-d**. The same preprocessing procedure will be applied to each dataset including the removal of cell types with fewer than 10 cells, the removal of genes expressed in less than 1% of cells in the RNA data modality, and the filtering of peaks quantified in less than 1% of cells in the ATAC data modality. All features will be retained for the ADT data modalities. For multiple ATAC data batches with different genomic ranges, we have followed https://stuartlab.org/signac/articles/merging using 'reduce' function from the *GenomicRanges* package and 'FeatureMatrix' function from the *Signac* package[13] to consolidate and quantify the intersecting peaks across all data batches to ensure a cohesive analysis of the data.

Data cleaning and reannotation

The reliance on predefined cell types can introduce bias towards methods that resemble those used to produce the original annotations. To minimise the potential bias in the original cell type annotation, we will conduct data cleaning and reannotation using AdaSampling[14] and Annotatability[15] techniques, followed by manual inspection for additional verification. In AdaSampling, a soft-label learning step is used to train a classification model for inferring mislabelling probability of each cell. Cells with high probabilities of mislabelling will be excluded in further iterations of model training. This iterative training process will enhance the accuracy of the classification model by systematically improving the quality of the cell type labels used for training. For each cell, the final trained model will be used to either confirm its original cell type annotation or identify potential mislabelling and possible correction for the cell. Similarly, Annotability refines cell type annotation with a deep neural network (DNN) by monitoring the confidence (quantified as the average probability of annotations) and the variability (quantified as the standard deviation for the probability of annotations) over training epochs. Cells are then categorised as correctly annotated, erroneously annotated, or ambiguous based on their confidence and variability. Possible corrections for cells in the erroneously annotated category are given by DNN trained solely on cells from the correctly annotated category.

The data cleaning and reannotation procedure is summarised in **Fig. 2e**. Specifically, we will reannotate cells to their newly nominated cell types if both AdaSampling and Annotability identified them as erroneously annotated and nominated them to the same new cell type. Cells that are identified as erroneously annotated by both methods but the newly annotated cell type disagree for the two methods will be removed. Finally, cells that are deemed as corrected annotated if one or both methods classify them to their original cell type. The final dataset after the cleaning and reannotation procedure will be manually validated by inspecting the expression of differentially expressed genes from each cell type and the annotated cells to verify if they are at a comparable level.

Simulation datasets

While the application of the above reannotation procedures may help reduce the potential cell annotation bias from the predefined cell types of each data source, simulation datasets can also be used as an alternative for method evaluation. We will use a generalised version of SymSim[16] to simulate single-cell multimodal omics for method evaluation. Specifically, we will simulate multiome data with RNA and ATAC modalities by following the tutorial (https://github.com/PeterZZQ/Symsim2) provided by the scDART[17] using the functions 'SimulateTrueCounts', 'True2ObservedCounts', and 'DivideBatches' in the package. For the simulation of ADT data modality, we follow the instruction outlined in scMoMaT[18]. This involves choosing a reference ADT count dataset to establish a count distribution, from which we then simulate ADT counts that exhibit a positive correlation with gene expression data. For integration methods that require gene activity scores as input, we simulate the gene activity score using the same procedure as in simulating the ADT modality. After simulating all paired modalities, we create various modality combinations according to the integration categories.

We have created a total of 22 simulated datasets. Details of each dataset grouped by integration categories are provided below.

Datasets for vertical integration

SD1: contains paired RNA and ADT data modalities of 3,000 cells from 3 cell types with 5,000 genes and 200 proteins.

SD2: contains paired RNA and ADT data modalities of 5,000 cells from 5 cell types with 3,000 genes and 200 proteins.

SD3: contains paired RNA and ATAC data modalities of 3,000 cells from 3 cell types with 2,000 genes and 3,000 peaks.

SD4: contains paired RNA and ATAC data modalities of 5,000 cells from 5 cell types with 1,000 genes and 2,000 peaks.

SD5: contains paired RNA, ADT, and ATAC data modalities of 3,000 cells from 3 cell types with 2,000 genes, 200 proteins, and 3,000 ATAC peaks.

SD6: contains paired RNA, ADT, and ATAC data modalities of 5,000 cells from 5 cell types with 1,000 genes, 200 proteins, and 2,000 ATAC peaks.

Datasets for diagonal integration

SD7: consists of unpaired RNA and ATAC data modalities. Both modalities contain 3,000 cells from 3 cell types. The RNA data modality contains 2,000 genes and the ATAC data modality contains 3,000 peaks that correspond to 2,000 gene activity scores.

SD8: consists of unpaired RNA and ATAC data modalities. Both modalities contain 5,000 cells from 5 cell types. The RNA data modality contains 1,000 genes and the ATAC data modality contains 2,000 peaks that correspond to 1,000 gene activity scores.

SD9: consists of 2 batches of unpaired RNA and ATAC data. The first batch contains 2,003 cells from 3 cell types and the second batch contains 1,997 cells from 3 cell types. For both batches, the RNA data modalities contain 2,000 genes and the ATAC data modalities contain 3,000 peaks that correspond to 2,000 gene activity scores.

SD10: consists of 2 batches of unpaired RNA and ATAC data. The first batch contains 2,514 cells from 3 cell types and the second batch contains 2,486 cells from 3 cell types. For both batches, the RNA data modalities contain 2,000 genes and the ATAC data modalities contain 5,000 peaks that correspond to 2,000 gene activity scores.

Datasets for mosaic integration

SD11: consists of 3 data batches where the first with only RNA modality (5,000 genes), the second with both RNA (5,000 genes) and ADT (200 proteins) modalities, and the third with only ADT modality (200 proteins). It contains 3,000 cells from 3 cell types for all 3 batches.

SD12: consists of 3 data batches where the first with only RNA modality (3,000 genes), the second with both RNA (3,000 genes) and ADT (200 proteins) modalities, and the third with only ADT modality (200 proteins). It contains 5,000 cells from 5 cell types for all 3 batches.

SD13: consists of 3 data batches where the first with only RNA modality (2,000 genes), the second with both RNA (2,000 genes) and ATAC (3,000 peaks) modalities, and the third with only ATAC modality (3,000 peaks). It contains 3,000 cells from 3 cell types for all 3 batches.

SD14: consists of 3 data batches where the first with only RNA modality (1,000 genes), the second with both RNA (1,000 genes) and ATAC (2,000 peaks) modalities, and the third with only ATAC modality (2,000 peaks). It contains 5,000 cells from 5 cell types for all 3 batches.

Datasets for cross integration

SD15: consists of 2 batches of CITE-seq data. The 2 batches contain 2,003 and 1,997 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes and 200 proteins.

SD16: consists of 2 batches of CITE-seq data. The 2 batches contain 2,514 and 2,486 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes and 200 proteins.

SD17: consists of 2 batches of paired RNA and ATAC data. The 2 batches contain 2,003 and 1,997 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes and 3,000 ATAC peaks.

SD18: consists of 2 batches of paired RNA and ATAC data. The 2 batches contain 2,514 and 2,486 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes and 5,000 ATAC peaks.

SD19: consists of 2 batches of paired ADT and ATAC data. The 2 batches contain 2,003 and 1,997 cells from 3 cell types, respectively. Both batches share the same set of features including 200 proteins and 3,000 ATAC peaks.

SD20: consists of 2 batches of paired ADT and ATAC data. The 2 batches contain 2,514 and 2,486 cells from 3 cell types, respectively. Both batches share the same set of features including 200 proteins and 5,000 ATAC peaks.

SD21: consists of 2 batches of paired RNA, ADT, and ATAC data. The 2 batches contain 2,003 and 1,997 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes, 200 proteins and 3,000 ATAC peaks.

SD22: consists of 2 batches of paired RNA, ADT, and ATAC data. The 2 batches contain 2,514 and 2,486 cells from 3 cell types, respectively. Both batches share the same set of features including 2,000 genes, 200 proteins and 5,000 ATAC peaks.

Single-cell multimodal omics data integration methods

In this section, we outline the single-cell multimodal omics data integration techniques included in this benchmark study. Count data is typically utilised as the input, unless otherwise indicated. For diagonal integration, we select methods specifically designed for aligning RNA and ATAC data modalities. Some methods require using original ATAC peaks directly as input, while others require peak data converted into gene activity scores. To facilitate this, we have implemented a cohesive pipeline, using the 'GeneActivity' function from the *Signac* package, to transform the ATAC matrix from peak level quantifications to gene activity scores. For diagonal integration, we use the gene activity scores from ATAC data as default unless indicated otherwise. For vertical, mosaic, and cross integration, we use the peak data of ATAC as default unless indicated otherwise. The appropriate data combinations and formats for each method are included in **Supplementary Table 2**. Details for each method are provided below.

M1: totalVI[19] ***(v1.1.2).*** We create the 'anndata' objects and employ the 'scvi.model.TOTALVI.setup_anndata', 'scvi.model.TOTALVI', and 'train' functions to construct train the totalVI model with default settings (https://docs.scvi-tools.org/en/stable/tutorials/notebooks/multimodal/totalVI.html). Finally, we employ the 'get_latent_representation' function to obtain the integrated embeddings. Besides, for the imputation purpose, we set 'include_protein_ background=True' in the 'get_normalized_expression' function to obtain the imputed ADT.

M2: sciPENN[20] ***(v1.0.0).*** We use 'sciPENN_API' and 'train' functions to build and train the model with the default parameters (https://github.com/jlakkis/sciPENN). sciPENN has the flexibility to train with or without labels; when labels are used, we specify 'type_key=cell_type' within the 'sciPENN_API' function. Finally, we use the 'embed' function to obtain the embeddings and use the 'predict' function to obtain the imputed modality.

M3: Concerto[21] ***(github version ab1fc7f).*** We construct the model using the 'multi_embedding_attention_transfer' function. The 'concerto_train_multimodal' function is then used to train the model with default settings (https://github.com/melobio/Concerto -reproducibility). Finally, we obtain the embeddings through the 'concerto_test_multimodal' function.

M4: scMSI[22] ***(github version dffcbb2).*** We construct the model using the 'SCMSIRNAProtein' function. Then, we train the model using the 'train' function with the default parameter settings (https://github.com/ChengmingZhang-CAS/scMSI-

master/tree/main). To obtain the integrated embeddings, we concatenate the embeddings from the RNA and ADT data.

M5: Matilda[23] ***(github version 7d71480).*** The count matrices along with the cell type labels, are used as input. As described in the authors' protocol (https://github.com/PYangLab/Matilda), we use 'main_matilda_train.py' to train the model with default parameters. Then, we use 'main_matilda_task.py' with 'dim_reduce=True' to obtain embeddings and set 'fs=True' to obtain the importance score for feature selection.

M6: MOFA+[24] ***(v1.6.0).*** As described in the vignette (https://raw.githack.com/bioFAM/MOFA2_tutorials/master/R_tutorials/getting_started_R.html), we use the 'create_mofa' function, with a list of batches to establish the MOFA object. Then, we train the model using the 'run_mofa' function, following the standard pipeline and parameters. Finally, we extract 'expectations$Z$group1' and 'intercepts$group1' from the resulting object to obtain the embeddings and importance score for feature selection, respectively.

M7: Multigrate[57] ***(v0.0.2).*** Following the authors' instructions (https://github.com/theislab/multigrate_reproducibility), we employ the 'data.organize_multiome_anndatas', 'model.MultiVAE.setup_anndata', 'model.MultiVAE', and 'train' functions with default settings to construct and train the model. Finally, we use the 'get_latent_representation' function to obtain the embeddings.

M8: UINMF[25] ***(v2.0.1).*** Following the authors' vignette (https://github.com/welch-lab/liger/blob/master/vignettes/articles/UINMF_vignette.Rmd), we apply functions such as 'createLiger', 'normalize', 'selectGenes', 'scaleNotCenter', and 'runIntegration' in the 'rliger' package to perform data integration with default settings. Finally, we extract 'H' from the resulting object to obtain the embeddings.

M9: scMoMaT[18] ***(v0.2.2).*** We first compile a list comprising multi-batch multi-modal data into the 'preprocess' function. Then, this data is fed into the 'scmomat_model' and 'train_func' to train the model with default settings (https://github.com/PeterZZQ/scMoMaT). Finally, we use 'extract_cell_factors' and 'calc_post_graph' functions to extract the embeddings, and the 'extract_marker_scores' function to obtain feature importance scores respectively.

M10: Seurat (WNN)[26] ***(v5.0.2).*** As described in the vignette (https://satijalab.org/seurat/articles/weighted_nearest_neighbor_analysis.html), we calculate the closest neighbours in the dataset using the 'FindMultiModalNeighbors' function with default settings. Finally, we extract 'weighted.nn' and 'wknn' from the resulting object to obtain the graphs.

M11: scMM[27] ***(github version c5c8579).*** We use the 'main.py' file to train the model to obtain the embeddings as described in their protocol (https://github.com/kodaim1115/scMM/tree/master). Specifically, we set the 'model' parameter based on the input data combination while keeping other parameters as default. Finally, we employ the 'get_latent' function to obtain the embeddings and use the 'predict' function to impute missing modalities.

M12: scMDC[28] ***(github version 43b0c3a).*** Following the authors' protocol (https://github.com/xianglin226/scMDC), we build the model using the 'scMultiClusterBatch' function with 'n_batch' as the actual number of batches. Then, the model is trained using the 'pretrain_autoencoder' function with the default parameters. Finally, we extract the embeddings through the 'encodeBatch' function."

M13: moETM[29] ***(github number c2eaa97).*** moETM is capable of being applied to bi-modality combinations such as RNA+ADT or RNA+ATAC. The count matrices from all modalities and the cell type labels are used as input. Following (https://github.com/manqizhou/moETM), we first preprocess the data, and then use 'build_moETM' and 'Train_moETM' functions to build and train the model with default settings. Finally, we extract 'delta' from the resulting object to obtain the embeddings. For imputation, we use the 'Trainer_moETM_for_cross_prediction' and 'Train_moETM_for_cross_prediction' function.

M14: VIMCCA[30] ***(v0.5.6).*** We use the 'log1p' and 'scale' function to process the input data, followed by the 'vimcca.fit_integration' function to train the model with default parameters (https://github.com/jhu99/scbean). The output from 'vimcca.fit_integration' is the integrated embedding.

M15: iPOLNG[31] ***(v0.0.2).*** The count matrices with 5,000 highly variable genes and 20,000 highly variable peaks are used as input (as author suggested). We use the 'iPoLNG' function with the default parameters to train the model (https://github.com/cuhklinlab/iPoLNG). Finally, we extract 'Ls_est' from the resulting object to obtain the embeddings.

M16: MIRA[32] ***(v2.1.0).*** We employ 'utils.make_joint_representation' to construct the joint representation space with default parameters (https://github.com/cistrome/MIRA). Finally, we calculate the integrated graph via the 'neighbours' function from the scanpy package on 'X_joint_umap_features' from the resulting AnnaData object.

M17: UnitedNet[33] ***(github version 3689da8).*** The count matrices along with the cell type labels, are used as input. We build the model using the 'UnitedNet' function. We adjust the 'input' dimension to match the actual dimension for the 'encoder', 'decoder', and 'discriminators', and set the 'output' in the 'clusters' to reflect the true number of cell types in the configuration. The model is then trained using the 'train' and 'finetune' functions with default settings (https://github.com/LiuLab-Bioelectronics-Harvard/UnitedNet). Subsequently, we use the 'infer' function to obtain the embeddings and 'predict' function to impute the missing modality in the test data.

M18: scMVP[34] ***(github version fc61e4d).*** The count matrices from paired RNA and ATAC are used as input, followed by standard preprocessing on both modalities. We build the model using the 'Multi_VAE_Attention' function. The model, which is then trained using the 'MultiTrainer' and 'train' functions with default settings (https://github.com/bm2-lab/scMVP). Subsequently, we use the 'create_posterior' function to create posterior from the trained model and 'get_latent' function to obtain the embeddings.

M19: scBridge[76] ***(github version ff17561).*** We use the 'Net' function and 'run' function to construct and train the network with the default parameters (https://github.com/XLearning-SCU/scBridge). Subsequently, we extract the 'Embedding' from the resulting object.

M20: Portal[35] ***(v1.0.2).*** We employ the 'model.Model' function to construct the model. Subsequently, we use the 'preprocess' function for data preparation, the 'train' function for model training, and the 'eval' function to obtain the integrated embeddings of the cells. All these steps are executed using default settings (https://github.com/YangLabHKUST/Portal).

M21: SCALEX[36] ***(v1.0.2).*** We use the 'SCALEX' function with the default parameters to train the model (https://github.com/jsxlei/SCALEX). Then, we extract the 'latent' from the resulting object to obtain the integrated embeddings.

M22: VIPCCA[37] ***(v0.2.7).*** We use the 'preprocessing' function to process the input data, followed by the 'VIPCCA' function to train the model with default parameters (https://github.com/jhu99/vipcca). Finally, we use the 'fit_integrate' function to obtain the integrated embedding.

M23: Seurat v3[38] ***(v5.0.2).*** We identify the anchors and co-embeddings using the 'VariableFeatures', 'GetAssayData', 'FindTransferAnchors', 'merge', 'ScaleData', and 'RunPCA' functions for integration (https://satijalab.org/seurat/articles/seurat5_atacseq_integration_vignettel). Specifically, the reduction parameter in the 'FindTransferAnchors' function is set to 'cca' by default. The integrated embeddings are extracted using the 'cell.embeddings' function.

M24: MultiMAP[39] ***(github version 681e608).*** The RNA count matrices and ATAC counts, which feature both peaks and gene activity scores, are utilised as inputs. Following the authors' recommendations (https://nbviewer.org/github/Teichlab/MultiMAP/blob/master/examples/tutorial.ipynb), we utilise the 'Integration' function to integrate the data with the default parameters. Then, we obtain the integrated graph from the .obsp['connectivities'] attribute.

M25: Seurat v5[40] ***(v5.0.2).*** Following their vignette (https://satijalab.org/seurat/articles/seurat5_integration_bridge.html), we utilise the 'PrepareBridgeReference', 'FindBridgeTransferAnchors', and 'MapQuery' functions to perform the integration. Finally, we extract 'cell.embeddings' to obtain the integrated embeddings.

M26: sciCAN[41] ***(github version ad71bba).*** The count matrices from both RNA and ATAC (gene activity scores) are used as inputs. Then, normalisation, log-transformation, and standard scaling are applied to both modalities. Following (https://github.com/rpmccordlab/sciCAN), we deploy the 'Cycle_train_wolabel' function with 200 epochs, while keeping other parameters as the default, to train our model. Then, we use the 'FeatureExtractor' function to extract the integrated embeddings from the model.

M27: Conos[42] ***(v1.5.2).*** Following the authors' instructions (https://htmlpreview.github.io/?https://raw.githubusercontent.com/kharchenkolab/conos/main/doc/integrating_rnaseq_atacseq.html), the count data from RNA and ATAC, presented as a list of matrices, is used as input for the Conos method. We perform pre-processing using the 'basicP2proc' function in the pagoda2 package. Then, we utilise functions such as 'new', 'buildGraph', 'findCommunities', and 'embedGraph' in the Conos package to perform the integration. Finally, we extract the 'graph_connectivities' to obtain the integrated graph.

M28: iNMF[43] ***(v2.0.1).*** We apply functions such as 'createLiger', 'normalize', 'selectGenes', 'scaleNotCenter', 'optimizeALS', and 'quantile_norm' to align the RNA and ATAC data with the default settings (https://github.com/welch-lab/liger/blob/master/vignettes/articles/Integrating_scRNA_and_scATAC_data.Rmd). Finally, we extract 'H.norm' to obtain the integrated embeddings.

M29: online iNMF[44] ***(v2.0.1).*** Following the authors' protocol (https://github.com/welch-lab/liger/blob/master/vignettes/articles/online_iNMF_tutorial.Rmd), we apply functions such as 'createLiger', 'normalize', 'selectGenes', 'scaleNotCenter', 'online_iNMF', and 'quantile_norm' with the default settings to integrate the RNA and ATAC data. Finally, we extract the embeddings in 'H.norm' to obtain the integrated embeddings.

M30: scJoint[45] ***(github version cbbfa5d).*** The count matrices along with the cell type labels, are used as input. We use the 'TrainingProcessStage1', 'KNN', and 'TrainingProcessStage3' functions to train the model as default (https://github.com/SydneyBioX/scJoint). Then, we use the encoder of the model to obtain the integrated embeddings.

M31: GLUE[46] ***(v0.3.2).*** The RNA count matrices and ATAC counts, which feature peaks, are utilised as inputs. Following their tutorial (https://github.com/gao-lab/GLUE), we construct a graph via the 'get_gene_annotation' function and build an RNA-anchored guidance graph with graph validation using the 'check_graph' function. Then, we use the 'configure_dataset' function to create a subgraph, and run the model using 'fit_SCGLUE'. All parameters are set to their default values as in the tutorial. Finally, we obtain the integrated embeddings using 'glue.encode_data'.

M32: SMILE[47] ***(github version a2e2ca6).*** The count matrices from the mosaic data combination of RNA and ATAC are used as input, where ATAC needs to be transformed into gene activity scores. Following (https://github.com/rpmccordlab/SMILE/blob/main/Tutorial/SMILE_data_integration_withReference.ipynb), we use 'littleSMILE' and 'ReferenceSMILE_trainer' functions the model. Upon completion of the training, we utilise the network's encoder to extract latent embeddings.

M33: StabMap[48] ***(v0.1.8).*** Following the vignette (https://marionilab.github.io/StabMap/articles/stabMap_PBMC_Multiome.html), a data list with multi-batch and multi-modal data is fed into the stabMap function using the default setting to do integration, resulting in the integrated embeddings. For imputation purposes, we use the 'imputeEmbedding' function to obtain the imputed data.

M34: MultiVI[49] ***(v1.1.2).*** We use the 'data.organize_multiome_anndatas', 'model.MULTIVI.setup_anndata', 'model.MULTIVI', and 'train' functions with the default settings to train the model (https://docs.scvi-tools.org/en/stable/tutorials/notebooks/multimodal/MultiVI_tutorial.html). Finally, we use the 'get_latent_representation' function to obtain the integrated embeddings. For imputation purposes, we use 'get_normalized_expression' and 'get_accessibility_estimates' functions, respectively, to obtain the imputed RNA and ATAC modalities.

M35: Cobolt*[50] *(v1.0.1). We use the 'MultiomicDataset.from_singledata' function to Create a MultiomicDataset from the given datasets. Then, we initialise and train the model using the 'Cobolt' (learning_rate = 0.0005) and 'train' functions with other parameters default (https://github.com/epurdom/cobolt). Finally, the integrated embeddings are acquired using the 'get_all_latent' function.

M36: PASTE*[51] *(v1.4.0). The count data with spatial coordinates are used as input. PASTE has two integration approaches (https://github.com/raphael-group/paste). The first is pairwise alignment, where we align pairwise slices using the 'pairwise_align function', followed by the 'stack_slices_pairwise' function to obtain the aligned coordinates. The second approach is centre alignment, for which we utilise the 'center_align' function to train the model and then obtain the aligned coordinates.

M37: SPIRAL*[52] *(v1.0). The count data with spatial coordinates are used as input. Following the their instructions (https://github.com/guott15/SPIRAL/blob/main/Demo/run_spiral_DLPFC.ipynb), we use the 'SPIRAL_integration.train', 'CoordAlignment.New_Coord' functions to train the model and then obtain the aligned coordinates.

M38: GPSA*[53] *(v0.8). The count data with spatial coordinates are used as input. Following their instructions (https://github.com/andrewcharlesjones/spatial-alignment), we use the 'VariationalGPSA' function to construct the model. We then perform a forward pass using the forward function and calculate the loss with 'loss_fn'. The network undergoes training for 100 epochs as default. Finally, we use the 'forward' function to obtain the aligned coordinates.

M39: PASTE2*[54] *(github version b71ec88). The count data with spatial coordinates are used as input. Following the authors' tutorial (https://github.com/raphael-group/paste2/blob/main/tutorial.ipynb), we align pairwise slices using 'partial_pairwise_align', followed by 'partial_stack_slices_pairwise' with default parameters to obtain the aligned coordinates.

Evaluation metrics

In this section, we outline the single-cell multimodal omics data integration evaluation metrics included in this benchmark study. For each metric, the advantages, disadvantages, requirements for cell type and batch labels, applicability to graph or embedding outputs, and other information is included in **Supplementary Table 3**. Summary for each metric is provided below.

Dimension reduction

Biological conservation metrics

Graph cell-type local inverse Simpson's index (Graph cLISI) score. The graph cell-type LISI (graph cLISI) score implemented in the *scib* package[12] assesses the cell-type separation of integrated data. It is calculated by extracting neighbourhood lists from integrated kNN graphs and computing graph distances using shortest path lengths. The graph cLISI scores are rescaled to a range of 0 to 1, quantifying the level of cell-type separation in the integrated data generated by a data integration method.

Dimension reduction scalability metrics

Since generating embeddings from single-cell multimodal omics data is a computationally intensive task and is also a critical component common to most integration methods, we benchmark methods for their scalability in terms of the computational time and memory usage required to complete this task.

Computational time on dimension reduction (CT). We evaluate running time on a server with AMD(R) Ryzen processor CPU (16 cores and 128 Gb total memory) and one RTX3090 graphics processing unit. For benchmarking purposes, we chose some datasets as our foundational datasets to create various benchmarking sets for measuring runtime. To evaluate the impact of the number of cells, we generate benchmarking datasets containing 500, 1,000, 2,000, 3,000, 5,000, 10,000, 20,000, 30,000, 50,000, 80,000, and 100,000 cells each. In cases where the base datasets consisted of multiple batches, each batch was configured to have an equal number of cells. To measure the elapsed runtime, we utilise the 'system.time()' function in R and the 'time()' function from the time package in Python, corresponding to methods implemented in R and Python, respectively.

Memory usage on dimension reduction (MU). Using the same benchmarking datasets as those employed for computing computational time, we adopt specific functions to monitor memory usage. For tracking CPU memory, we use the '/usr/bin/time -v command' for R and Python. Meanwhile, GPU memory consumption is monitored using the 'nvidia-smi' tool. To calculate the total memory usage involving both CPU and GPU, we sum the memory readings from both sources.

Batch correction

Batch correction quality metrics

K-nearest neighbour batch effect test (kBET) score. The kBET score quantifies the similarity between the label composition of a k-nearest neighbourhood in an integrated dataset and the expected label composition. It was introduced to measure the rejection rate of tested neighbourhoods in k-nearest neighbour (kNN) graphs to evaluate batch effects. We will apply the kBET algorithm (v.0.99.6, release 4c9dafa) implemented in the *scib* package[12] to quantify the batch effect.

Graph integration LISI score (iLISI). Similar to graph cLISI score but calculated for batches in the integrated data, the graph integration LISI (iLISI) score assesses the batch mixing of integrated data. The graph iLISI scores are rescaled to a range of 0 to 1, quantifying the level of batch integration achieved by the data integration methods.

Adjusted Rand Index by batch (ARI_batch). ARI is a metric used to assess the concordance between two partitions of a set of items by considering both the overlaps and the disagreements between the two partitions[55]. Here ARI_batch will be used for quantifying the concordance between cells grouped by clusters and by their batch labels in a dataset.

Normalised mutual information by batch (NMI_batch). Similar to ARI, NMI is a metric used to assess the agreement between two partitions of a set of items and ranges between 0 (uncorrelated) and 1 (perfect match).

Average Silhouette width by batch (ASW_batch). The silhouette width quantifies the within-cluster distances of a cell and the between-cluster distances of that cell to the closest cluster[56]. Averaging the silhouette width across all cells gives ASW. Luecken et al.[12] proposed a modified ASW for quantifying batch mixing ('ASW_batch'). We will follow this definition and its implementation in the *scib* package[12] for calculating ASW_batch where a value of 1 represents ideal batch mixing and 0 indicates strong batch separation.

Graph connectivity (GC). The GC metric assesses the connectedness of cells with the same cell-type label in the integrated kNN graph. It is a metric introduced by Luecken et al.[12] for quantifying batch effect. We will use its implementation in the *scib* package[12] for calculating GC.

Principal component regression (PCR) score. The PCR score is used to assess the impact of data integration on variance, comparing the explained variance before and after integration. It is a metric implemented in *scib* package[12], which ranges from 0 to 1, where 0 indicates no change in variance contribution, and 1 indicates a significant difference in variance contributions before and after integration.

Clustering

Clustering metrics

Adjusted Rand Index by cell type (ARI_cellType). ARI_cellType is used for quantifying the concordance between cells grouped by clusters and by cell type annotation in their original publications. The definition of ARI_cellType is the same as ARI_batch except that the batch labels are replaced by cell type labels.

Normalised mutual information by cell type (NMI_cellType). NMI_cellType is used for quantifying the concordance between cells grouped by clusters and by cell type annotation in their original publications. The definition of NMI_cellType is the same as NMI_batch except that the batch labels are replaced by cell type labels.

Average silhouette width by cellType (ASW_cellType). As described above, the silhouette width quantifies the within-cluster distances of a cell and the between-cluster distances of that cell to the closest cluster and its average across all cells gives ASW. It can be calculated from the *scib* package[12] for evaluating the separation of cell types (i.e., ASW_cellType).

Isolated label score of F1 (iF1). iF1 is defined as the labels that appear in the minimum number of batches. This score introduced by Luecken et al.[12] indicates how well the isolated labels are separated from other cell types and can be calculated from the *scib* package[12] for quantifying the quality of clustering results.

Isolated label score ASW (isolated_labels_asw). The *scib* package[12] calculates the ASW for the isolated labels in comparison to the non-isolated labels. This score is then scaled to a range between 0 and 1. We average the isolated scores for all isolated labels and present the final average score.

Classification

Classification metrics

Overall classification accuracy (OCA). Overall classification accuracy is determined by dividing the number of correctly predicted cells by the total number of cells in the dataset. This metric provides an overall measure of a model's predictive performance in correctly classifying cells.

Average classification accuracy (ACA). Average classification accuracy is computed by summing up the accuracies across each cell type and then dividing by the number of cell types in the dataset. This metric is particularly useful in datasets where the number of cells between cell types significantly varies, ensuring that the model's performance is evaluated across all cell types, rather than being influenced by the performance of more abundant cell types.

Specificity (Spec) and sensitivity (Sens). Specificity measures the proportion of actual negatives that are correctly identified. A higher specificity indicates better detection of true negatives, reducing the chance of false positives. Sensitivity is defined as the proportion of true positives correctly identified in the test. It quantifies how effective a model is to identify positive cases among all the truly positive cases. The range for both specificity and sensitivity is from 0 to 1, with values closer to 1 indicating better performance. We will use one-versus-all to calculate the specificity and sensitivity for the classification performance of a method for each cell type.

F1-score (F1). F1-score is computed as the harmonic mean of precision and recall. Precision is the ratio of true positives within the predicted positive cases and recall is equal to sensitivity. The F1-score ranges from 0 to 1, where 0 denotes the poorest performance and 1 represents the best performance. We will use one-versus-all to calculate the F1-score for the classification performance of a method for each cell type.

Feature selection

Feature selection is frequently used for identifying cell type markers and can be used in various downstream analyses. We will select top markers by their ranking and/or feature importance scores with respect to each cell type determined by each feature selection method.

Feature specificity and reproducibility metrics

Marker overlap among different cell types (MO).
We identify the top 5, 10, and 20 markers for different cell types and assess the overlaps among these markers via the *intersect* function in R. A smaller overlap suggests the markers selected for each cell type are distinct and specific.

Marker correlation among downsampled data (MC). We randomly subsample cells to various percentages of the original data (100%) to 80%, 50%, 30%, and 10%. We then run the feature selection methods from the targeted integration approaches to derive the feature importance scores for each subsampled dataset and calculate the Pearson's correlation of feature importance scores across different sampling results. A higher correlation indicates higher feature selection reproducibility.

Feature selection downstream analysis metrics

Clustering metrics applied to top features. We take the union of the top 5, 10, and 20 markers from each cell type, as identified from each feature selection method, to conduct clustering analysis. The metrics used in this analysis will be the same as those employed in the clustering task, i.e., ARI_cellType, NMI_cellType, ASW_cellType and iF1, isolated_labels_asw.

Classification metrics applied to top features. We take the union of the top 5, 10, and 20 markers from each cell type, as identified from each feature selection method, to conduct classification analysis. The metrics used in this analysis will be the same as those employed in the classification task, i.e., OCA, ACA, Spec, Sens and F1.

Imputation

Imputation data structure metrics

MSE imputation error with ground truth data (MSE). This metric is calculated by taking the mean squared error (MSE) of the differences between the imputed data and the ground truth data heldout from the model training. A lower MSE imputation error indicates that the imputed data is closer to the ground truth data, reflecting higher accuracy in the imputation process.

Preservation of feature correlation structure (pFCS). As implemented in the Matilda framework[23], we will calculate the pairwise correlation of the top 100 highly variable features in the imputed data and ground truth data heldout from the model training. The preservation of gene correlation structure can then be quantified by calculating Pearson's correlation of the pairwise correlations computed from the imputed data and ground truth data, with a higher correlation indicating better imputation performance.

Preservation of differential expression statistics (pDES). We will apply the Limma R package[58] to calculate differential expression (DE) statistics of features for each cell type and compute the average Pearson's correlation of DE statistics calculated from the imputed data and the ground truth data. A higher correlation indicates better preservation of DE statistics in the imputed data.

Imputation downstream analysis metrics

Clustering metrics applied to imputed data. As illustrated in Fig. 1d, we will conduct clustering analysis on the imputed data. The metrics used in this analysis will be the same as those employed in the clustering task, i.e., ARI_cellType, NMI_cellType, ASW_cellType and iF1, isolated_labels_asw.

Classification metrics applied to imputed data. As illustrated in Fig. 1d, we will conduct classification analysis on the imputed data. The metrics used in this analysis will be the same as those employed in the classification task, i.e., OCA, ACA, Spec, Sens and F1.

Spatial registration

Spatial registration quality metrics

Label Transfer Adjusted Rand Index (LTARI). As described in Liu et al.[54], LTARI is a metric used to evaluate the spatial registration performance in preserving cell type and/or spatial region labels between the aligned spots across slices. First, we compute the clustering result for each slice. Then, one slice is selected as the reference, and the others are treated as query slices. For the query data, LTARI assigns new labels to each spot according to the label

of the nearest spot in the reference slice. Finally, we calculate the ARI between the newly assigned spot labels of the query slice and the ground truth labelling, thus assessing the registration performance in maintaining label consistency across slices.

Pairwise Alignment Accuracy (PAA). As implemented in PASTE[51], PAA is calculated by summing the weighted pairs of spots that share the same cluster labels across all slices. Then, we average the PAA for all paired slices and present the final average score. A higher PAA indicates better spatial registration performance.

Spatial Coherence Score (SCS). Following the implementation in PASTE[51], the SCS is calculated by employing O'Neill's spatial entropy to quantify the variation in spatial coherence of cluster labels across different slices. Higher SCS values suggest a patterned label distribution among adjacent spots, while lower values indicate a distribution that is closer to random. The average SCS for all slices is calculated to present the final SCS.

Spatial registration scalability metrics

Spatial registration is another computationally expensive task and the benchmark of method scalability will provide useful information about their usability in this task. Given that spatial registration methods do not aim to generate integrated embeddings or graphs, they are not included in previous scalability benchmarks. Consequently, they require the following independent comparison methods to assess their scalability.

Computational time on spatial registration (CT_SR). For benchmarking spatial registration methods, we sample from the spatial dataset to create various datasets with different cell numbers (i.e., 500, 1,000, 2,000, 3,000, 5,000, 10,000, 20,000, 30,000, 50,000, 80,000, and 100,000) for measuring runtime. Each batch was configured to have an equal number of cells. All configurations and functions applied align with those detailed in the dimension reduction metrics section.

Memory usage on spatial registration (MU_SR). Using the same sets of datasets in the above benchmark of computational time for spatial registration methods, we will also evaluate and benchmark the memory usage using the same functions previously outlined in the dimension reduction metrics section.

Evaluation pipelines

The following sections describe the flowchart and pseudocode for the seven task evaluation pipelines. The details of the datasets, integration methods, tasks and evaluation metrics involved are described in the corresponding sections above. The aligned initial code for each pipeline is provided at https://github.com/PYangLab/scMultiBench/tree/main/metrics.

Dimension reduction evaluation pipeline

The dimension reduction task evaluation pipeline is summarised in **Fig. 3**. The input data can be either single-batch multimodal count data or multi-batch multimodal count data, and the multimodal integration methods typically generate low-dimensional embeddings that capture the multimodal characteristics of individual cells. The quality of the embeddings generated by the integration methods can be evaluated using the Graph cLISI score. We will also assess the scalability of these integration methods by analysing the computational time (CT) and memory usage (MU).

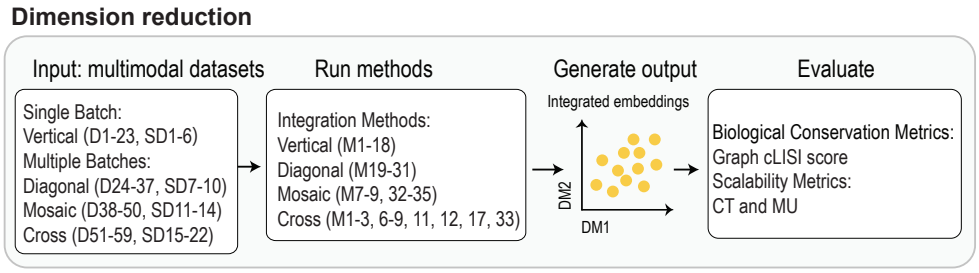

Fig. 3: Schematic summary of the evaluation pipelines for the integration methods on the dimension reduction task.

Algorithm 1 below provides the pseudocode detailing the steps for evaluating each integration method in this task. Specifically, for each targeted integration method ***m***, we first prepare the list of corresponding single-batch or multi-batch dataset files ***D*** and generate the integrated embeddings which is denoted by $\boldsymbol{embedding_i}$ for each dataset file $\boldsymbol{d_i}$ (Line 5-7). Then, we calculate the 3 evaluation metrics $\boldsymbol{e_j}$, including sLISI score, CT and MU, for each $\boldsymbol{embedding_i}$ to generate the evaluation result list ***O*** and compare across different integration methods (Line 9-12). The list of generated embeddings ***B*** will be saved to files for later use (Line 8). Specifically, with our dimension reduction evaluation pipeline, we will analyse 18 vertical integration methods on 23 real datasets and 6 simulated datasets, 13 diagonal integration methods on 14 real datasets and 4 simulated datasets, 7 mosaic integration methods on 13 real datasets and 4 simulated datasets and 11 cross integration methods on 9 real datasets and 8 simulated datasets (**Fig. 3**).

Algorithm 1: Pseudocode for Evaluating Dimension Reduction

```
1  Define
       inputs: list of dataset paths D; list of evaluation metrics E;
               integration method m;
       output: list of integrated embeddings B; list of evaluation results
               O;
2  EvaluateDimensionReduction (D, E, m)
3      B ← list();
4      O ← list();
5      for data path d_i ∈ D do
6          data_i ← load_multimodal_data(d_i);
7          embedding_i ← generate_embedding(m, data_i);
8          append(B, embedding_i);
9          for evaluation metrics e_j ∈ E do
10             result_ij ← evaluate(embedding_i, e_j);
11             append(O, result_ij);
12         end
13     end
14 return B, O;
```

Batch correction evaluation pipeline

Batch correction will be carried out using various multi-batch integration methods, including 13 diagonal integration methods on 14 real datasets and 4 simulated datasets, 7 mosaic integration methods on 13 real datasets and 4 simulated datasets and 11 cross integration methods on 9 real datasets and 8 simulated datasets. These methods can generate batch-corrected low-dimensional embeddings, which will be evaluated using the 7 batch correction metrics. **Fig. 4** illustrates the evaluation pipeline for the batch correction task. Similar to the dimension reduction evaluation, we first derive the batch-corrected integrated embeddings for each targeted method and on the corresponding datasets. Then, we apply the 7 batch correction metrics to all the embeddings for comparison across different methods. The evaluation pipeline for each method follows the steps in **Algorithm 1**, with the only difference that the evaluation function ***EvaluateBatchCorrection(D, E, m)*** will be used with the evaluation metrics ***E*** for the 7 batch correction metrics.

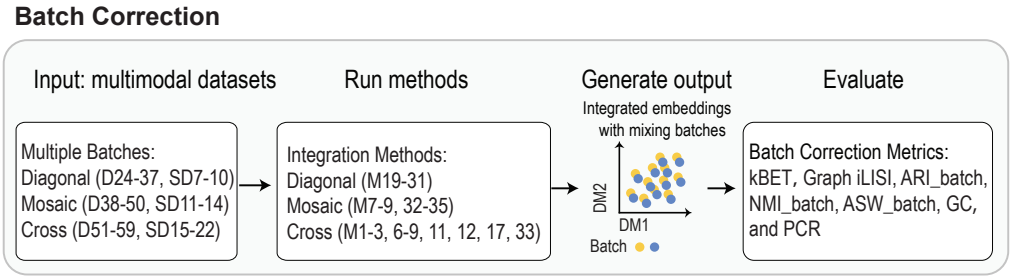

Fig. 4: Schematic summary of the evaluation pipelines for the integration methods on the batch correction task.

Clustering evaluation pipeline

The clustering evaluation pipeline is summarised in **Fig. 5**. In particular, clustering will be performed on the low-dimensional integrated embeddings generated from the dimension reduction methods using Leiden clustering within the *scib* package[12]. All integration methods and datasets from the dimension reduction evaluation pipeline will be assessed, including 18 vertical integration methods on 23 real datasets and 6 simulated datasets, 13 diagonal integration methods on 14 real datasets and 4 simulated datasets, 7 mosaic integration methods on 13 real datasets and 4 simulated datasets and 11 cross integration methods on 9 real datasets and 8 simulated datasets. The quality of the clustering results will be measured using the 5 clustering evaluation metrics, including ARI_cellType, NMI_cellType, ASW_cellType, iF1 and isolated_labels_asw.

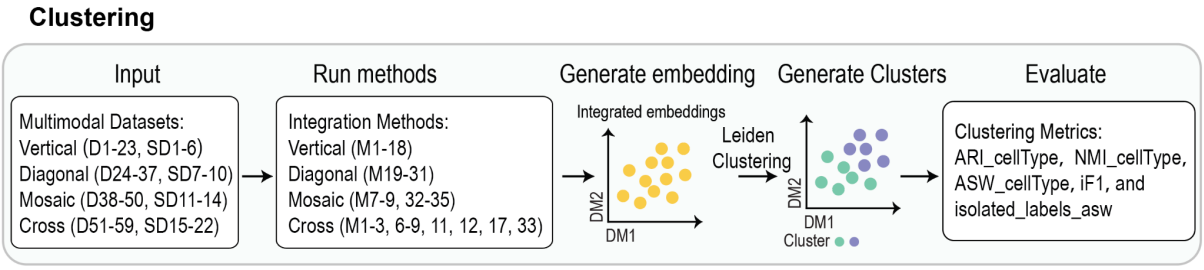

Fig. 5: Schematic summary of the evaluation pipelines for the integration methods on the clustering task.

Algorithm 2 below details the pseudocode for the steps involved in evaluating each integration method on the clustering task. We first load the pre-saved embeddings from a method ***m*** from the corresponding file paths ***F*** (Line 4-5). For each ***embedding$_i$***, we apply the Leiden clustering to derive the ***clusters$_i$*** (Line 6). Then, we evaluate the clustering results based on each quality metric (Line 7-10). The evaluation output ***O*** will be used for our comparative analyses across different methods and datasets.

Algorithm 2: Pseudocode for Evaluating Clustering

```
1  Define
       inputs : list of evaluation metrics E; list of embeddings files F
                generated via integration method m;
       output: list of evaluation results O;
2  EvaluateClustering (F, E)
3      O ← list();
4      for embedding path f_i ∈ F do
5          embedding_i ← load_integrated_embedding(f_i);
6          clusters_i ← leiden_clustering(embedding_i);
7          for evaluation metrics e_j ∈ E do
8              result_ij ← evaluate(clusters_i, embedding_i, e_j);
9              append(O, result_ij);
10         end
11     end
12 return O;
```

Classification evaluation pipeline

Fig. 6 summarises the workflow of the classification evaluation pipeline. Similar to the batch correction pipeline above, classification is also performed and evaluated on the integrated low-dimensional embeddings generated from the dimension reduction methods, with the focus on the multi-batch integration methods, including the 13 diagonal integration methods on 14 real datasets and 4 simulated datasets, the 7 mosaic integration methods on 13 real datasets and 4 simulated datasets and the 11 cross integration methods on 9 real datasets and 8 simulated datasets. All datasets included in this task contain multiple batches and for each dataset and a specific method, each data batch is used as a reference, with the remaining batches as queries to assess prediction results using a simple linear neural network (NN) classifier. If a method also provides its own classification model, we will also include the classification results using its own model. Finally, we summarise the results from across all classification combinations for each dataset and method.

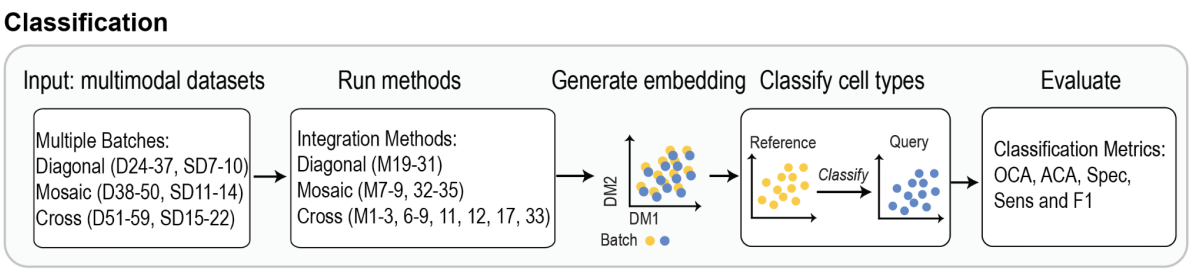

Fig. 6: Schematic summary of the evaluation pipelines for the integration methods on the Classification task.

Algorithm 3 provides the pseudocode for the steps involved in evaluating each integration method. Specifically, for each pre-save ***embedding$_i$*** of the method ***m***, we construct a data structure named ***classification_summary*** for saving the classification evaluation summary for each integration method and dataset (Line 5-7). We first iterate over each data batch of the embeddings as reference (Line 8-9) while using the remaining batches of embeddings as queries (Line 10). Then, for each query, we apply the classification function that calls each specific method to derive the classification results for the current reference data (Line 11-12). The classification results ***predictions*** are then evaluated using each of the metrics in ***E*** (i.e., OCA, ACA, Spec, Sens and F1) and stored in the ***classification_summary*** structure for the final evaluation analysis (Line 13-16).

Algorithm 3: Pseudocode for Evaluating Classification

```
1  Define
       inputs : list of evaluation metrics E; list of embeddings files F generated via
                integration method m;
       output: list of evaluation results O;
2  EvaluateClassification (F, E, m)
3      O ← list();
4      for embedding path f_i ∈ F do
5          embedding_i ← load_integrated_embedding(f_i);
6          number_of_batches ← get_number_of_batches(embedding_i);
7          classification_summary ← initialize_classification_summary();
8          foreach batch_id in 1,2,..., number_of_batches do
9              reference ← select_batch_as_reference(embedding_i, batch_id);
10             queries ← select_batches_as_queries(embedding_i, batch_id);
11             foreach query in queries do
12                 predictions ← classify_cells(m, reference, query);
13                 for evaluation metrics e_j ∈ E do
14                     evaluation_score ← evaluate_classification(predictions, e_j);
15                     update_summary(classification_summary, evaluation_score);
16                 end
17             end
18         end
19         append(O, classification_summary);
20     end
21 return O;
```

Feature selection evaluation pipeline

Summarised in the evaluation pipeline in **Fig. 7**, we will focus on the 3 integration methods that enable feature selection, including Matilda, scMoMaT and MOFA+. The quality of feature selection is evaluated on the 23 real datasets and 6 simulated datasets and using the aforementioned 5 classification metrics, 5 clustering metrics, and the feature specificity and reproducibility metrics (i.e., MO and MC). In particular, we will train a sample linear NN classifier and conduct 5-fold cross-validation for evaluating the utility of selected features for classifying cell types and will also apply the Leiden clustering pipeline as in the clustering task to evaluate their utility in cell clustering.

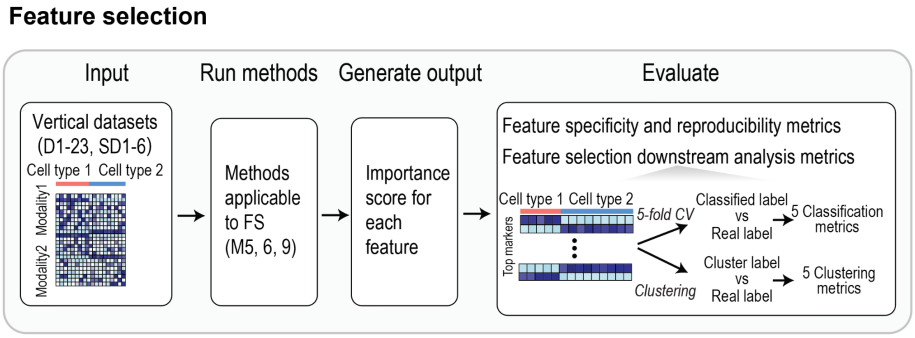

Fig. 7: Schematic summary of the evaluation pipelines for the integration methods on the feature selection task.

Algorithm 4 below provides the pseudocode for the steps involved in the feature selection for each integration method. For each integration method ***m*** on each dataset $\boldsymbol{d_i}$, we first run the method to generate the feature importance scores (Line 6). For evaluating feature specificity and their utility on classification and clustering, we select the top 5, 10 and 20 features for each cell type based on the feature importance scores and calculate the evaluation results based on the corresponding metrics (Line 10-18). Particularly, for classification, ***classification_cross_validation(top_features,*** $\boldsymbol{E_{cls}}$***)*** in Line 14 will conduct the 5-fold cross-validation via a linear NN classifier using the union of selected top features and then assess the performance by using the 5 classification evaluation metrics. For clustering, ***evaluate_clustering_leiden(top_features,*** $\boldsymbol{E_{clu}}$***)*** in Line 16 will apply the Leiden clustering pipeline and calculate the 5 clustering evaluation metrics. Finally, for feature reproducibility, we will first randomly subsample 10%, 30%, 50%, 80% of cells and run the integration method to produce the corresponding feature importance scores for each subsample (Line 20-24). We will then calculate the pairwise Pearson's correlation for each subsample (Line 25). All the evaluation results will be saved for comparative analysis (Line 31).

Algorithm 4: Pseudocode for Evaluating Feature Selection

```
1  Define
       inputs : list of dataset paths D; integration method m; feature specificity evaluation
                metrics e_s; feature reproducibility evaluation metrics e_r, list of classification
                evaluation metrics E_cls, list of clustering evaluation metrics E_clu;
       output: list of feature specificity evaluation results O_s, feature reproducibility
                evaluation results O_r, classification evaluation results O_cls, clustering
                evaluation results O_clu;
2  EvaluateFeatureSelection (D, e_s, e_r, E_cls, E_clu, m)
3      O_s, O_r, O_cls, O_clu ← list();
4      for data path d_i ∈ D do
5          data_i ← load_multimodal_data(d_i);
6          importance_score ← apply_feature_selection(data_i, m);
7          specificity_results ← list();
8          classification_results ← list();
9          clustering_results ← list();
10         for top_k ∈ {5, 10, 20} do
11             top_features ← select_top_features(importance_score, top_k);
12             specificity ← evaluate_specificity(top_features, e_s);
13             append(specificity_results, specificity);
14             classification ← classification_cross_validation(top_features, E_cls);
15             append(classification_results, classification);
16             clustering ← evaluate_clustering_leiden(top_features, E_clu);
17             append(clustering_results, clustering);
18         end
19         importance_score_subs ← list();
20         for portion ∈ {0.1, 0.3, 0.5, 0.8} do
21             subset_data ← apply_random_subset(data_i, portion);
22             importance_score_for_sub ← apply_feature_selection(subset_data , m);
23             append(importance_score_subs, importance_score_for_sub);
24         end
25         reproducibility_results ← evaluate_reproducibility(importance_score_subs, e_r);
26         append(O_s, specificity_results);
27         append(O_r, reproducibility_results);
28         append(O_cls, classification_results);
29         append(O_clu, clustering_results);
30     end
31 return O_s, O_r, O_cls, O_clu;
```

Imputation evaluation pipeline

Fig. 8 illustrates the imputation evaluation pipeline. We will focus on the 7 mosaic integration methods applicable to imputation. This evaluation involves using reference and query data generated from datasets D51-57. In particular, we will randomly select one batch as the reference data and then select another batch as query but only keep one of data modality in the query dataset. This allows the missing data modality in the query data to be used as the ground truth for evaluating the quality of the imputed data. During the imputation, we utilise the reference data to impute the missing data modalities in the query data and then compare the imputed data with the ground truth data. The quality of the imputed data is quantified using 3 imputation data structure metrics (i.e., MSE, pFCS, pDES) as well as downstream analysis metric, including 5 classification metrics and 5 clustering metrics. Specifically, for the classification assessment, we will first train a simple NN classifier using the reference data subset by the features that are included in the imputed data (denoted by * in **Fig. 8**) and then classify the cells using the imputed data. This approach allows for the assessment of imputed data using the 5 classification metrics.

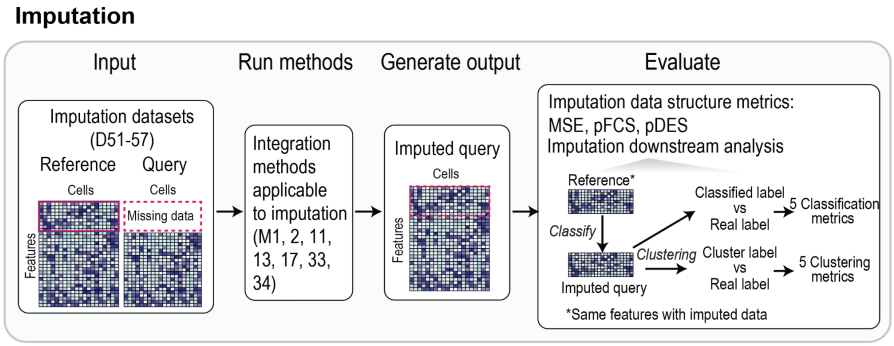

Fig. 8: Schematic summary of the evaluation pipelines for the integration methods on the imputation task.

Algorithm 5 below provides the corresponding pseudocode detailing the steps for evaluating each integration method on the imputation task. After loading the prepared dataset (Line 5), we first randomly select one batch as the reference data (Line 7-8) while selecting other batches as query data (Line 9). Then, for each batch in the query data, we exclude one modality to create the query data with missing features (Line 10-11) and keep the excluded data as ground truth for evaluation (Line 12). We run the integration method to impute the missing modality in the query data based on the reference data (Line 13). With the derived imputed data, we then evaluate the quality using the 3 imputation data structure metrics (Line 14-17), the 5 classification metrics (Line 19-22) and the 5 clustering metrics (24-27). Specifically, the ***label_transfer()*** in Line 18 denotes the aforementioned cell classification of the query data using the reference data, whereas the ***leiden_clustering()*** in Line 23 indicates the leiden clustering pipeline as introduced in the evaluation procedure of previous tasks. All the evaluation results are saved for the comparative analysis.

Algorithm 5: Pseudocode for Evaluating Imputation

```
Define
    inputs : list of dataset paths D; integration method m; list of imputation data structure
             evaluation metrics E_imp, list of classification evaluation metrics E_cls, list of clustering
             evaluation metrics E_clu;
    output: list of evaluation results O
EvaluateImputation (D, E_imp, E_cls, E_clu, m)
    O ← list();
    for data path d_i ∈ D do
        data_i ← load_multimodal_data(d_i);
        evaluation_for_dataset ← list();
        batch_id ← select_random_batch(data_i);
        reference ← get_batch_as_reference(data_i, batch_id);
        queries ← get_batches_as_queries(data_i, batch_id);
        foreach query in queries do
            query_with_missing_data ← create_query(query);
            missing_data_groud_truth ← create_query_ground_truth(query);
            imputed_data ← impute(reference, query_with_missing_data, m);
            for evaluation metrics e_j ∈ E_imp do
                evaluation_result ← evaluate_imputation(imputed_data,
                  missing_data_groud_truth, e_j);
                append(evaluation_for_dataset, evaluation_result);
            end
            classifications ← label_transfer(imputed_data, missing_data_groud_truth);
            for evaluation metrics e_k ∈ E_cls do
                evaluation_result ← evaluate_classification(classifications, e_k);
                append(evaluation_for_dataset, evaluation_result);
            end
            clusters ← leiden_clustering(imputed_data);
            for evaluation metrics e_m ∈ E_clu do
                evaluation_result ← evaluate_clustering(clusters, e_m);
                append(evaluation_for_dataset, evaluation_result);
            end
        end
        append(O, evaluation_for_dataset);
    end
return O;
```

Spatial registration evaluation pipeline

The spatial registration evaluation pipeline focuses on the 4 integration methods applicable to this task using the 5 spatial datasets with multiple slices, with each spatial slice consisting of the gene expression and spatial coordinates. **Fig. 9** illustrates the evaluation schematic which uses multiple spatial slices as the input. After conducting spatial registration via the integration methods, these slices are aligned into a consistent coordinate system (i.e., aligned coordinates). For evaluation, we employ 3 established metrics from existing literature, including LTARI, PAA, and SCS. For LTARI, we perform leiden clustering on all slices independently to obtain cluster labels for each slice. We will next choose one slice as the reference, while the remaining as queries and transfer labels from the reference to the queries using 1-nearest neighbours based on the aligned coordinates. Finally, we will apply the ARI metric to the transferred label and ground truth to calculate LTARI. For PAA and SCS, they are calculated based on the reference labels and the labels transferred to the query data. Besides, we will also apply 2 spatial registration scalability metrics for the evaluation, including CT_SR and MU_SR. As described in the evaluation metrics section, we will randomly sample from dataset 53 to create 11 sub-datasets with different cell numbers ranging from 500 to 100,000, based on which the runtime (CT_SR) and memory usage (MU_SR) will be evaluated.

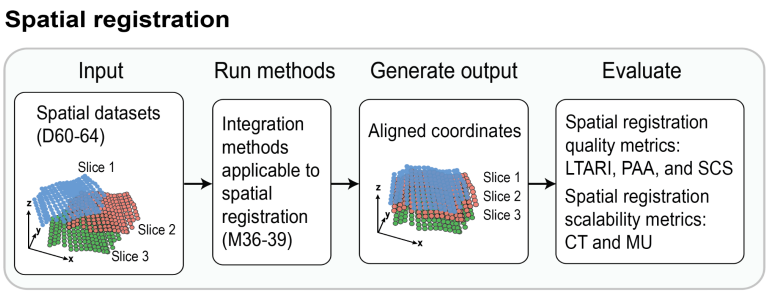

Fig. 9: Schematic summary of the evaluation pipelines for the integration methods on the spatial registration task.

Algorithm 6 provides the corresponding pseudocode detailing the steps for evaluating each integration method on the spatial registration task. For each loaded spatial dataset ***data$_i$*** with multiple slices, we will run the integration method to generate the registered slices ***registered_slices*** in the aligned coordinates (Line 5-7). Then, we will apply the leiden clustering on each registered slice with regard to both gene expression and spatial coordinates (Line 9-12). As is described above, we will utilise the label transfer from the reference to the query slice to evaluate the 3 spatial registration quality metrics (Line 14-17). More specifically, the ***evaluate_quality()*** function in Line 15 includes the label transfer procedure for each reference slice and its corresponding query slices. For the evaluation of scalability (Line 20-23), the ***evaluate_sclability()*** function in Line 21 will apply each scalability metrics to the aforementioned sub-datasets to measure both the runtime and memory. All the evaluation results will be saved for our benchmarking analysis.

Algorithm 6: Pseudocode for Evaluating Spatial Registration

1 **Define**
  inputs : list of spatial dataset paths D; spatial integration method m; list of spatial registration quality metrics E_q; list of spatial registration scalability metrics E_s
  output: list of evaluation results O
2 **EvaluateSpatialRegistration** (D, E_q, E_s, m)
3 $O \leftarrow$ list();
4 $evaluation_for_dataset \leftarrow$ list();
5 **for** *data path* $d_i \in D$ **do**
6 $data_i \leftarrow$ load_spatial_data(d_i);
7 $registered_slices \leftarrow$ perform_spatial_registration($data_i$, m);
8 $cluster_labels \leftarrow$ list();
9 **foreach** $slice_rna$, $slice_coordinates$ ***in*** $registered_slices$ **do**
10 $cluster_labels_for_slice \leftarrow$ clustering($slice_rna$, $slice_coordinates$);
11 append($cluster_labels$, $cluster_labels_for_slice$);
12 **end**
13 $quality_evaluation_results \leftarrow$ list();
14 **for** *evaluation metrics* $e_j \in E_q$ **do**
15 $evaluation_result \leftarrow$ evaluate_quality($data_i$, $registered_slices$, $cluster_labels$, e_j);
16 append($quality_evaluation_results$, $evaluation_result$);
17 **end**
18 append($evaluation_for_dataset$, $quality_evaluation_results$);
19 $scalability_evaluatio_results \leftarrow$ list();
20 **for** *evaluation metrics* $e_k \in E_s$ **do**
21 $evaluation_result \leftarrow$ evaluate_scalability($data_i$, $registered_slices$, $cluster_labels$, e_k);
22 append($scalability_evaluation_results$, $evaluation_result$);
23 **end**
24 append($evaluation_for_dataset$, $scalability_evaluation_results$);
25 append(O, $evaluation_for_dataset$);
26 **end**
27 **return** O;

Other benchmark considerations

Robustness and consistency assessment

We will randomly remove a subset of cell types (~20%) from datasets to assess the robustness of methods on data perturbation. The random removal of cell types will be repeated 10 times and for each iteration, we will evaluate the performance of the integration using metrics for their respective tasks. The change of performance on perturbed datasets will be used to assess the robustness of each method on all integration categories and tasks. We will also assess the impact of methods that rely on random initialisation by randomising initial values/seeds and repeating these methods (10 times) to evaluate their integration performance using metrics for their respective tasks. Finally, we will assess the consistency of the methods by grouping and ranking their performance results by technologies and/or platforms. This will allow us to quantify the change of their performance across different technologies and platforms.

Clustering algorithms and parameters

We will use popular clustering algorithms of k-means, Leiden, and Louvain, for tasks that require clustering to evaluate the performance by different. Specifically, for k-means, we will set k, the number of clusters, to the actual number of cell types in each dataset. For Leiden and Louvain clustering, we will employ the 'get_N_clusters' function from the Sinfonia package[59] to automatically select the resolution parameters that align the number of clusters to the number of cell types in each dataset. This will allow us to quantify the performance of integration methods across different clustering algorithms.

R Shiny application for interactive exploration of benchmark results

We will create an R Shiny application to host the benchmark results for interactive exploration, enabling the users to select specific integration categories, tasks, evaluation metrics, and datasets of interest and dynamically display the relative performance of methods across the selections. The application will also enable the inclusion of new methods and datasets as they become available, ensuring up-to-date communication of benchmark results.

Guideline and Recommendations

We will provide a decision-tree-style guideline for each integration and task category as the final recommendation for method selection. Specifically, the decision-tree-style guideline will include scenario-specific recommendations for methods, taking into consideration data characteristics and computational efficiency. Each branch of the decision tree will display three recommended methods tailored to the specific integration category and task. These methods will be accompanied by key performance metrics benchmarked in this study, including overall performance, performance in specific tasks, and speed, and represented using colour bars. The structured guidelines will facilitate clear and comprehensive method recommendations for users.

Timeline

We expect a 5-month timeline for completing the study proposed in Stage 1. The summary of the plan is as follows:

- Month 1-2.5: Obtain all results for dimension reduction, batch correction, clustering, and classification tasks, and perform visualisation and analysis.
- Month 2.5-5: Obtain all results for feature selection, imputation, and spatial registration tasks, and perform visualisation and analysis, and write the manuscript.

Data availability

Published data were downloaded and used to compile the 64 datasets described in the section "Datasets and preprocessing" for benchmarking methods in each integration category and for different tasks. The source data information is detailed in **Supplementary Table 4**. Below is a brief summary of each data source.

Data source 1: CITE-seq data capturing RNA and ADT modalities of human peripheral blood mononuclear cell (PBMC) samples[60]. Cell types in the original study were manually annotated based on RNA and protein expression of known markers from each cell cluster. This annotation was further verified by matching to the Azimuth PBMC database for quality validation using the Azimuth tool (http://azimuth.satijalab.org/app/azimuth). We obtained the data under the accession of E-MTAB-10026 from the Array Express database.

Data source 2: CITE-seq data capturing RNA and ADT modalities of human PBMC samples[61]. Cell types in the original study were annotated using cluster specific differential expression and the reference-based scRNA-seq annotation method implemented in SingleR[77] (v1.4.0). We obtained the data under the accession of GSE166489 from the GEO database.

Data source 3: CITE-seq data capturing RNA and ADT modalities of human PBMC samples[26]. Cell types in the original study were manually annotated based on RNA and protein expression of both known markers and those discovered through clustering analyses. We obtained the data under the accession of GSE164378 from the GEO database.

Data source 4: CITE-seq data capturing RNA and ADT modalities of human bone marrow mononuclear cell (BMMC)[62]. Cell types in the original study were annotated first by known markers from each cluster generated by either RNA or ADT modality using the Scanpy platform, and then, manually validated and cross-modality harmonised. We obtained the data under the accession of GSE194122 from the GEO database.

Data source 5: CITE-seq data capturing RNA and ADT modalities of human BMMC samples[38]. Cell types in the original study were annotated via mapping protein expression from the CITE-seq data to a Human Cell Atlas reference dataset and transferring the labels via Seurat v3. We obtained the data under the accession of https://zenodo.org/records/6348128 from the Zenodo database.

Data source 6: CITE-seq data capturing RNA and ADT modalities of human PBMC samples and lung tissues[63]. Cell types in the original study were manually annotated based on cluster-specific expression of markers within the RNA and/or ADT modality and further aided by cell-by-cell annotation from the SingleR (v1.4.0) using the 'Monaco reference' from the celldex (v1.0.0) made from bulk RNA-seq data of sorted immune cell populations. We obtained the data from https://archive.softwareheritage.org/browse/revision/1c7fcabb18a1971dc4d6e29bc3ed4f6f36b2361f/.

Data source 7: SHARE-seq data capturing RNA and ATAC modalities of mouse skin samples[5]. Cell types in the original study were manually assigned to clusters on the basis of marker genes, TF motifs, and chromatin accessibility peaks. We obtained the data under the accession of GSE140203 from the GEO database.

Data source 8: SNARE-seq data capturing RNA and ATAC modalities of mouse brain samples[64]. Cell types in the original study were manually annotated on the basis of known markers for the cerebral cortex and gene expression patterns from DropViz (http://dropviz.org/). We obtained the data under the accession of GSE126074 from the GEO database.

Data source 9: 10x multiome data capturing RNA and ATAC modalities of human BMMC samples[62]. Cell types in the original study were first determined by cluster-related markers from RNA and ATAC modalities independently using the Scanpy platform. Specifically, the count matrix of open chromatin data was first converted to a gene activity matrix and were then used for the marker-based cell type annotation. Next, modality-specific cell types were manually harmonised. We obtained the data under the accession of GSE194122 from the GEO database.

Data source 10: 10x multiome data capturing RNA and ATAC modalities of mouse embryo samples[65]. Cell types in the original study were annotated by mapping the RNA expression profiles to a reference atlas from the same developmental stages. Specifically, the query and atlas cells were first integrated within a joint PCA space. Then, each query cell was assigned with a cell type based on majority voting over the cell types from its 25 nearest atlas neighbours. We obtained the data under the accession of GSE205117 from the GEO database.

Data source 11: 10x multiome data capturing RNA and ATAC modalities of human PBMC samples. Cell type annotations in the original study were first performed by 10x Genomics and then further validated manually by inspecting markers from an annotated single-cell RNA atlas. We obtained the validated data from https://zenodo.org/records/6348128.

Data source 12: 10x multiome data capturing RNA and ATAC modalities of human neocortex samples[66]. Cell types in the original study were manually annotated on the basis of marker genes for each cluster. We obtained the data under the accession of GSE204684 from the GEO database.

Data source 13: ASAP-seq data capturing ATAC and ADT modalities of human PBMC samples[67]. Cell types in the original study were manually annotated based on the marker gene expression in each cluster in the RNA and ADT modalities. We obtained the data under the accession of GSE156478 from the GEO database.

Data source 14: DOGMA-seq data capturing RNA, ATAC, and ADT modalities of human PBMC samples[68]. Cell types in the original study were annotated by projecting the three-modality WNN (3WNN) clusters derived from Seurat to Azimuth PBMC reference data. We obtained the data under the accession of GSE200417 from the GEO database.

Data source 15: TEA-seq data capturing RNA, ATAC, and ADT modalities of human PBMC samples[6]. Cell types in the original study were derived via label transfer from a multimodal reference atlas for PBMCs using Seurat. We obtained the data under the accession of GSE158013 from the GEO database.

Data source 16: Visium spatial data capturing RNA and spatial coordinates of mouse squamous cell carcinoma[69]. Spot labels in the original study were generated based on independent component analysis. Data were obtained from https://zenodo.org/records/6334774

Data source 17: Visium spatial data capturing RNA and spatial coordinates of human dorsolateral prefrontal cortex samples[70]. Spot labels in the original study were manually annotated based on cytoarchitecture visualised by spatialLIBD tool and selected gene markers. Data were obtained from https://github.com/raphael-group/paste_reproducibility/tree/main/data/DLPFC.

Data source 18: Xenium spatial data capturing RNA and spatial coordinates of human breast cancer tumour. Cell types in the original study were manually annotated based on differential gene expression from each cluster[71]. Data were obtained from https://www.10xgenomics.com/products/xenium-in-situ/preview-dataset-human-breast

Data source 19: Stereo-seq spatial data capturing RNA and spatial coordinates of drosophila embryos[72]. Tissue types at bin level in the original study were manually annotated based on marker genes for each cluster. The clustering was performed using both transcriptomic information and spatial information (via Squidpy[73]). Data were obtained from https://db.cngb.org/stomics/flysta3d/

Data source 20: MERFISH spatial data capturing RNA and spatial coordinates of mouse brain[74]. Cell types of the original study were manually annotated based on the comparison of marker genes and/or spatial locations of clusters, which were obtained from the multi-level clustering of integrated MERFISH and snRNA-seq data, with previously annotated datasets. Data were obtained from https://cellxgene.cziscience.com/collections/31937775-0602-4e52-a799-b6acdd2bac2e

Data source 21: Spatial ATAC-RNA seq data capturing RNA, ATAC and spatial coordinates of mouse embryo (E13) and mouse postnatal (P22) brain[75]. In the original study, cell types in spatial-RNA were transferred from scRNA-seq using Seurat while cell types in spatial-ATAC were transferred from scATAC-seq data using Seurat and Signac. We obtained the data under the accession of GSE205055 from the GEO database.

Code availability

All source code for all published results from this study will be made available at https://github.com/PYangLab/scMultiBench

Competing interests

The authors declare they have no competing interests.

Authors contribution

CL, HJK, and PY conceived the study. CL and PY drafted the manuscript with input for all authors. CL, SL, SD, and DX performed the initial feasibility analysis of the project. CL will lead the benchmark study with anticipated major contribution from SL, HJK, SD, and PY,

and additional contribution from DX and SG. All authors read, edited, and approved this work.

Acknowledgements

The authors thank all their colleagues, particularly at the School of Mathematics and Statistics, The University of Sydney, and Sydney Precision Data Science Centre for their support and intellectual engagement. This work is supported by a National Health and Medical Research Council (NHMRC) Investigator Grant (1173469) to P.Y; and an ARC DECRA (DE220100964) to S.G.

References

1. Baysoy, A., Bai, Z., Satija, R. & Fan, R. The technological landscape and applications of single-cell multi-omics. *Nat. Rev. Mol. Cell Biol.* **24**, 695–713 (2023).
2. Vandereyken, K., Sifrim, A., Thienpont, B. & Voet, T. Methods and applications for single-cell and spatial multi-omics. *Nat. Rev. Genet.* **24**, 494–515 (2023).
3. Gaulton, K. J., Preissl, S. & Ren, B. Interpreting non-coding disease-associated human variants using single-cell epigenomics. *Nat. Rev. Genet.* **24**, 516–534 (2023).
4. Stoeckius, M. *et al.* Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods* **14**, 865–868 (2017).
5. Ma, S. *et al.* Chromatin Potential Identified by Shared Single-Cell Profiling of RNA and Chromatin. *Cell* **183**, 1103-1116.e20 (2020).
6. Swanson, E. *et al.* Simultaneous trimodal single-cell measurement of transcripts, epitopes, and chromatin accessibility using TEA-seq. *eLife* **10**, e63632 (2021).
7. Argelaguet, R., Cuomo, A. S. E., Stegle, O. & Marioni, J. C. Computational principles and challenges in single-cell data integration. *Nat. Biotechnol.* **39**, 1202–1215 (2021).
8. Heumos, L. *et al.* Best practices for single-cell analysis across modalities. *Nat. Rev. Genet.* **24**, 550–572 (2023).
9. Miao, Z., Humphreys, B. D., McMahon, A. P. & Kim, J. Multi-omics integration in the age of million single-cell data. *Nat. Rev. Nephrol.* **17**, 710–724 (2021).
10. Cantini, L. *et al.* Benchmarking joint multi-omics dimensionality reduction approaches

for the study of cancer. *Nat. Commun.* **12**, 124 (2021).

11. Leng, D. *et al.* A benchmark study of deep learning-based multi-omics data fusion methods for cancer. *Genome Biol.* **23**, 171 (2022).
12. Luecken, M. D. *et al.* Benchmarking atlas-level data integration in single-cell genomics. *Nat. Methods* **19**, 41–50 (2022).
13. Stuart, T., Srivastava, A., Madad, S., Lareau, C. A. & Satija, R. Single-cell chromatin state analysis with Signac. *Nat. Methods* **18**, 1333–1341 (2021).
14. Yang, P. *et al.* AdaSampling for Positive-Unlabeled and Label Noise Learning With Bioinformatics Applications. *IEEE Trans. Cybern.* **49**, 1932–1943 (2019).
15. Karin, J., Mintz, R., Raveh, B. & Nitzan, M. Interpreting single-cell and spatial omics data using deep networks training dynamics. 2024.04.06.588373 Preprint at https://doi.org/10.1101/2024.04.06.588373 (2024).
16. Zhang, X., Xu, C. & Yosef, N. Simulating multiple faceted variability in single cell RNA sequencing. *Nat. Commun.* **10**, 2611 (2019).
17. Zhang, Z., Yang, C. & Zhang, X. scDART: integrating unmatched scRNA-seq and scATAC-seq data and learning cross-modality relationship simultaneously. *Genome Biol.* **23**, 139 (2022).
18. Zhang, Z. *et al.* scMoMaT jointly performs single cell mosaic integration and multi-modal bio-marker detection. *Nat. Commun.* **14**, 384 (2023).
19. Gayoso, A. *et al.* Joint probabilistic modeling of single-cell multi-omic data with totalVI. *Nat. Methods* **18**, 272–282 (2021).
20. Lakkis, J. *et al.* A multi-use deep learning method for CITE-seq and single-cell RNA-seq data integration with cell surface protein prediction and imputation. *Nat. Mach. Intell.* **4**, 940–952 (2022).
21. Yang, M. *et al.* Contrastive learning enables rapid mapping to multimodal single-cell

atlas of multimillion scale. *Nat. Mach. Intell.* **4**, 696–709 (2022).

22. Zhang, C. *et al.* Contrastively generative self-expression model for single-cell and spatial multimodal data. *Brief. Bioinform.* **24**, bbad265 (2023).
23. Liu, C., Huang, H. & Yang, P. Multi-task learning from multimodal single-cell omics with Matilda. *Nucleic Acids Res.* **51**, (2023).
24. Argelaguet, R. *et al.* MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data. *Genome Biol.* **21**, 111 (2020).
25. Kriebel, A. R. & Welch, J. D. UINMF performs mosaic integration of single-cell multi-omic datasets using nonnegative matrix factorization. *Nat. Commun.* **13**, 780 (2022).
26. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *Cell* **184**, 3573-3587.e29 (2021).
27. Minoura, K., Abe, K., Nam, H., Nishikawa, H. & Shimamura, T. scMM: Mixture-of-experts multimodal deep generative model for single-cell multiomics data analysis. 2021.02.18.431907 Preprint at https://doi.org/10.1101/2021.02.18.431907 (2021).
28. Lin, X., Tian, T., Wei, Z. & Hakonarson, H. Clustering of single-cell multi-omics data with a multimodal deep learning method. *Nat. Commun.* **13**, 7705 (2022).
29. Zhou, M. *et al.* Single-cell multi-omics topic embedding reveals cell-type-specific and COVID-19 severity-related immune signatures. *Cell Rep. Methods* **3**, 100563 (2023).
30. Wang, Y. *et al.* A multi-view latent variable model reveals cellular heterogeneity in complex tissues for paired multimodal single-cell data. *Bioinformatics* **39**, btad005 (2023).
31. Zhang, W. & Lin, Z. iPoLNG—An unsupervised model for the integrative analysis of single-cell multiomics data. *Front. Genet.* **14**, (2023).
32. Lynch, A. W. *et al.* MIRA: joint regulatory modeling of multimodal expression and chromatin accessibility in single cells. *Nat. Methods* **19**, 1097–1108 (2022).

33. Tang, X. *et al.* Explainable multi-task learning for multi-modality biological data analysis. *Nat. Commun.* **14**, 2546 (2023).

34. Li, G. *et al.* A deep generative model for multi-view profiling of single-cell RNA-seq and ATAC-seq data. *Genome Biol.* **23**, 20 (2022).

35. Zhao, J. *et al.* Adversarial domain translation networks for integrating large-scale atlas-level single-cell datasets. *Nat. Comput. Sci.* **2**, 317–330 (2022).

36. Xiong, L. *et al.* Online single-cell data integration through projecting heterogeneous datasets into a common cell-embedding space. *Nat. Commun.* **13**, 6118 (2022).

37. Hu, J., Chen, M. & Zhou, X. Effective and scalable single-cell data alignment with non-linear canonical correlation analysis. *Nucleic Acids Res.* **50**, e21–e21 (2022).

38. Stuart, T. *et al.* Comprehensive Integration of Single-Cell Data. *Cell* **177**, 1888-1902.e21 (2019).

39. Jain, M. S. *et al.* MultiMAP: dimensionality reduction and integration of multimodal data. *Genome Biol.* **22**, 346 (2021).

40. Hao, Y. *et al.* Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nat. Biotechnol.* 1–12 (2023) doi:10.1038/s41587-023-01767-y.

41. Xu, Y., Begoli, E. & McCord, R. P. sciCAN: single-cell chromatin accessibility and gene expression data integration via cycle-consistent adversarial network. *Npj Syst. Biol. Appl.* **8**, 1–10 (2022).

42. Barkas, N. *et al.* Joint analysis of heterogeneous single-cell RNA-seq dataset collections. *Nat. Methods* **16**, 695–698 (2019).

43. Welch, J. D. *et al.* Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity. *Cell* **177**, 1873-1887.e17 (2019).

44. Gao, C. *et al.* Iterative single-cell multi-omic integration using online learning. *Nat. Biotechnol.* **39**, 1000–1007 (2021).

45. Lin, Y. *et al.* scJoint integrates atlas-scale single-cell RNA-seq and ATAC-seq data with transfer learning. *Nat. Biotechnol.* **40**, 703–710 (2022).

46. Cao, Z.-J. & Gao, G. Multi-omics single-cell data integration and regulatory inference with graph-linked embedding. *Nat. Biotechnol.* **40**, 1458–1466 (2022).

47. Xu, Y., Das, P. & McCord, R. P. SMILE: mutual information learning for integration of single-cell omics data. *Bioinformatics* **38**, 476–486 (2022).

48. Ghazanfar, S., Guibentif, C. & Marioni, J. C. Stabilized mosaic single-cell data integration using unshared features. *Nat. Biotechnol.* 1–9 (2023) doi:10.1038/s41587-023-01766-z.

49. Ashuach, T. *et al.* MultiVI: deep generative model for the integration of multimodal data. *Nat. Methods* **20**, 1222–1231 (2023).

50. Gong, B., Zhou, Y. & Purdom, E. Cobolt: integrative analysis of multimodal single-cell sequencing data. *Genome Biol.* **22**, 351 (2021).

51. Zeira, R., Land, M., Strzalkowski, A. & Raphael, B. J. Alignment and integration of spatial transcriptomics data. *Nat. Methods* **19**, 567–575 (2022).

52. Guo, T. *et al.* SPIRAL: integrating and aligning spatially resolved transcriptomics data across different experiments, conditions, and technologies. *Genome Biol.* **24**, 241 (2023).

53. Jones, A., Townes, F. W., Li, D. & Engelhardt, B. E. Alignment of spatial genomics data using deep Gaussian processes. *Nat. Methods* **20**, 1379–1387 (2023).

54. Liu, X., Zeira, R. & Raphael, B. J. PASTE2: Partial Alignment of Multi-slice Spatially Resolved Transcriptomics Data. *bioRxiv* 2023.01.08.523162 (2023) doi:10.1101/2023.01.08.523162.

55. Rand, W. M. Objective Criteria for the Evaluation of Clustering Methods. *J. Am. Stat. Assoc.* **66**, 846–850 (1971).

56. Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of

cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987).

57. Lotfollahi, M., Litinetskaya, A. & Theis, F. J. Multigrate: single-cell multi-omic data integration. 2022.03.16.484643 Preprint at https://doi.org/10.1101/2022.03.16.484643 (2022).
58. Ritchie, M. E. *et al.* limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Res.* **43**, e47–e47 (2015).
59. Jiang, R., Li, Z., Jia, Y., Li, S. & Chen, S. SINFONIA: Scalable Identification of Spatially Variable Genes for Deciphering Spatial Domains. *Cells* **12**, 604 (2023).
60. Stephenson, E. *et al.* Single-cell multi-omics analysis of the immune response in COVID-19. *Nat. Med.* **27**, 904–916 (2021).
61. Ramaswamy, A. *et al.* Immune dysregulation and autoreactivity correlate with disease severity in SARS-CoV-2-associated multisystem inflammatory syndrome in children. *Immunity* **54**, 1083-1095.e7 (2021).
62. Luecken, M. D. *et al.* A sandbox for prediction and integration of DNA, RNA, and proteins in single cells. in (2021).
63. Buus, T. B. *et al.* Improving oligo-conjugated antibody signal in multimodal single-cell analysis. *eLife* **10**, e61973 (2021).
64. Chen, S., Lake, B. B. & Zhang, K. High-throughput sequencing of the transcriptome and chromatin accessibility in the same cell. *Nat. Biotechnol.* **37**, 1452–1457 (2019).
65. Argelaguet, R. *et al.* Decoding gene regulation in the mouse embryo using single-cell multi-omics. 2022.06.15.496239 Preprint at https://doi.org/10.1101/2022.06.15.496239 (2022).
66. Zhu, K. *et al.* Multi-omic profiling of the developing human cerebral cortex at the single-cell level. *Sci. Adv.* **9**, eadg3754 (2023).
67. Mimitou, E. P. *et al.* Scalable, multimodal profiling of chromatin accessibility, gene

expression and protein levels in single cells. *Nat. Biotechnol.* **39**, 1246–1258 (2021).

68. Xu, Z., Heidrich-O'Hare, E., Chen, W. & Duerr, R. H. Comprehensive benchmarking of CITE-seq versus DOGMA-seq single cell multimodal omics. *Genome Biol.* **23**, 135 (2022).
69. Ji, A. L. *et al.* Multimodal Analysis of Composition and Spatial Architecture in Human Squamous Cell Carcinoma. *Cell* **182**, 497-514.e22 (2020).
70. Maynard, K. R. *et al.* Transcriptome-scale spatial gene expression in the human dorsolateral prefrontal cortex. *Nat. Neurosci.* **24**, 425–436 (2021).
71. Janesick, A. *et al.* High resolution mapping of the tumor microenvironment using integrated single-cell, spatial and in situ analysis. *Nat. Commun.* **14**, 8353 (2023).
72. Wang, M. *et al.* High-resolution 3D spatiotemporal transcriptomic maps of developing *Drosophila* embryos and larvae. *Dev. Cell* **57**, 1271-1283.e4 (2022).
73. Palla, G. *et al.* Squidpy: a scalable framework for spatial omics analysis. *Nat. Methods* **19**, 171–178 (2022).
74. Allen, W. E., Blosser, T. R., Sullivan, Z. A., Dulac, C. & Zhuang, X. Molecular and spatial signatures of mouse brain aging at single-cell resolution. *Cell* **186**, 194-208.e18 (2023).
75. Zhang, D. *et al.* Spatial epigenome–transcriptome co-profiling of mammalian tissues. *Nature* **616**, 113–122 (2023).
76. Li, Y. *et al.* scBridge embraces cell heterogeneity in single-cell RNA-seq and ATAC-seq data integration. *Nat. Commu*n. 14, 6045 (2023).
77. Aran, D. *et al.* Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nat. Immunol.* 20, 163–172 (2019).