A Appendix

A.1 Annotation of the benchmark dataset

A.1.1 Raw data processing

Raw read data from 10X multiome and CITE-seq data was processed using the CZ Biohub alignment pipeline available at `https://github.com/czbiohub/utilities/tree/neevor/cellrangerarc`. Both pipelines were run using AWS batch and with reference refdata-cellranger-arc-GRCh38-2020-A-2.0.0.tar.gz provided by 10X. For ATAC-seq plus GEX, cellranger-arc-2.0.0 was used to run cellranger-arc count on each individual sample. Then all samples were run with cellranger-arc aggr to produce the final multiomics dataset. For CITE-seq plus GEX, cellranger-6.1.0 was used to run cellranger count on each sample. Internal steps of the pipeline used pandas v1.3.1, numpy v1.21.1, and scanpy v1.8.1. All pipelines were built using docker v20.10.7 and deployed to AWS ERC for use with AWS BATCH.

A.1.2 Gene expression data

Gene expression data from the 10X Multiome (nuclear data) and CITE-seq (whole cell data) protocols were both analyzed using our previously published best practices [20]. We used the Scanpy platform [38] as a basis for quality control, normalization, dimensionality reduction, clustering, feature selection, and trajectory inference.

Quality control of cellular data was performed per sample by thresholding the number of molecular counts (UMIs) per cell and the number of genes per cell. Considering the joint distribution of these quantities, we selected minimum thresholds ranging from 300-750 and 280-750, respectively, per sample. Furthermore, an upper threshold on UMI counts between 22,000 and 38,000 was selected also on a per-sample basis. Genes with observations in fewer than 20 cells per sample were removed from the dataset.

To enable comparisons between cellular expression profiles that may have received different numbers of reads during sequencing, we normalized the data. Normalization was performed by the pooling method implemented in the *computeSumFactors()* function in Scran v1.20.1 [41]. To improve the signal-to-noise, we selected 4000 highly variable genes (HVGs) as implemented by the "cell ranger" method in Scanpy. Here, highly variable genes are selected by binning genes by mean expression and choosing the genes with the highest coefficient of variation per bin. We used the first 50 principal components of the HVG-subsetted expression matrix as a low dimensional representation of the data. To apply graph-based visualization and clustering algorithms to the data, we generated a k-nearest neighbour (kNN) graph using Euclidean distance on the PC space as implemented in Scanpy. The data was then visualized using the UMAP algorithm [42] and clustered by Leiden community detection [43] v0.8.7 at a range of resolutions. We finally extracted cluster-related features using pairwise t-tests over the cluster assignments per cluster and compared these to published literature on bone marrow mononuclear cells.

A.1.3 Open chromatin data

The chromatin accessibility data acquired by ATAC-seq as part of the 10X Multiome protocol was processed using Signac v1.3.0 [39], an extension of the Seurat toolkit v4.0.3 [16], and the Scanpy platform v1.7.2 [38]. To ensure the same set of features across samples, accessible regions (also referred to as peaks) were aggregated using *cellranger-arc aggr*. Quality control, dimensionality reduction and translating peaks to gene activity scores was performed using Signac, following the authors' instructions. Downstream analysis steps including cell type annotation and trajectory inference were done in Scanpy.

After loading the peak-by-cell matrix, counts were binarized to only represent an accessible versus non-accessible state of each region. Cells were then filtered based on 5 quality control metrics comprising the total number of fragments (ranging from 200-850 to 60,000-150,000 across samples), the enrichment of fragments detected at transcription start sites (TSS) (ranging from 2.2-4.1 to 10.5-20 across sample), the fraction of fragments in peak regions compared to peak-flanking regions (lower limit between 0.2-0.455 across samples), the fraction of peaks blacklisted by the ENCODE consortium [44] (upper limit ranging between of 0.0075-0.015 across samples) and the nucleosome

signal, which describes the length distribution of fragments which is expected to follow the length of DNA required span across one nucleosome or multiples of it (upper limit ranging from 2-2.5 across samples). Since ATAC data is sparser than gene expression data, peaks were included if they were accessible in at least 15 cells.

Dimensionality reduction was performed by generating term frequencies using latent semantic indexing (LSI) initially suggested by Cusanovich et al. [45], followed by singular value decomposition. LSI components with a high correlation (absolute value > 0.51) with the total number of fragments per cell were removed prior to subsequent analysis steps. Visualisation, clustering and cell type annotation was performed as described in the gene expression data analysis with the difference of using LSI components as the low dimensional representation of the data. Since peaks only refer to regions in the genome, they are difficult to interpret directly. Therefore, the count matrix was translated to a gene activity matrix by summing up accessible regions over the gene bodies including promoter regions (defined as 2kb upstream of the TSS). These gene activity scores were used for a marker-based cell type annotation.

A.1.4 Protein data

The workflow of analyzing cell surface protein levels captured as antibody-derived tags (ADT) in the CITE-seq protocol was adapted from our pipeline to process gene expression data and mainly performed using the Scanpy platform v1.7.2 [38]. The TotalSeq-B antibody panel from BioLegend Inc. used in this study comprises 134 primary antibodies capturing human cell surface proteins and 6 isotype controls without any human target protein that can be used to assess the level of unspecific binding in each cell.

Quality control was done based on the total number of ADTs (ranging from 1100-1200 to 24000 across samples), the number of proteins captured in each cell (with a lower limit of 80) and the ADT count of the 6 isotype controls summed up in each cell (ranging from 1 to 100). Since the total number of captured ADTs is limited, absolute ADT counts appear to be lower if highly abundant proteins are present. To account for this effect, normalization was performed using the centered log ratio transformation implemented in the *NormalizeData()* in Seurat v4.0.3 [16]. Dimensionality reduction, computation of a k-nearest neighbour (kNN) graph, clustering and visualisation was performed analogously to the gene expression data analysis. Cell surface protein markers derived from the literature were used for cell type annotation.

A.1.5 Harmonizing cell labels between joint modalities

Following modality- and batch-specific data analysis, we harmonized the cell type annotation per batch by taking the outer product of the cluster annotation to ensure substructure present in only one modality was still preserved in the final annotations. Where cluster substructure did not agree and did not lead to a clean subclustering, we manually evaluated which modality marker features more clearly described the specific cellular subpopulation.

A.1.6 Annotating trajectories in the data

To capture continuous cellular, we inferred and annotated the erythrocyte differentiation trajectory. This trajectory goes from hematopoietic stem cells (HSCs) via megakaryocyte and erythrocyte progenitors (MK/E prog), proerythroblasts, and erythroblasts, to normoblasts and reticulocytes (if present in the data) in the bone marrow. Using a similar approach as in [7], we subsetted the relevant clusters and fitted trajectory to the data in diffusion map space using the diffusion pseudotime algorithm [28]; implemented in Scanpy v1.7.2. In brief, this method runs a diffusion process on the single-cell kNN graph and embeds the data into a spectral decomposition of the obtained transition matrix. A linear trajectory is described by a so-called pseudotime ordering of cells, which is computed based on the distance to a root cell in diffusion space. The root cell was manually determined as a cell in the HSC cluster with an extremal embedding in the first two diffusion components.

To ensure that our ground-truth trajectories were not affected by a particular embedding of the data, trajectories were fit separately per modality and batch. Here, the uni-modal kNN graph representations generated separately from each modality were used as a basis for trajectory inference.

A.2 Joint embedding metrics

Performance in task 3 will be measured using seven metrics broken into two classes:

- Biological variance conservation
- Batch correction

These measures are then aggregated into a single score used to rank embedding methods.

A.2.1 Bio-conservation metrics

These metrics measure how well an embedding reflects expert-annotated biology.

1. **NMI cluster/label** - Normalized mutual information (NMI) compares the overlap of two clusterings. We use NMI to compare the cell type labels with an automated clustering computed on the integrated dataset (based on Louvain clustering). NMI scores of 0 or 1 correspond to uncorrelated clustering or a perfect match, respectively. Automated Louvain clustering is performed at resolution ranges from 0.1 to 2 in steps of 0.1, and the clustering output with the highest NMI with the label set is used.
2. **ARI cluster/label** - The Rand index compares the overlap of two clusterings; it counts both correct clustering overlaps and correct disagreements between two clusterings. Similar to NMI, we compare the cell type labels with the NMI-optimized Louvain clustering computed on the integrated dataset. An ARI of 0 or 1 corresponds to random labelling or a perfect match, respectively.
3. **Cell type ASW** - The silhouette width measures the compactness of observations with the same labels. Averaging over all silhouette widths of a set of cells yields the average silhouette width (ASW), which ranges between -1 and 1. We use ASW to evaluate the compactness of cell types in the resulting embedding. The cluster ASW is computed on cell identity labels and scaled to a value between 0 and 1 using the equation:

   $$ASW = (ASW_C + 1)/2$$

   where C denotes the set of all cell identity labels.
4. **Cell cycle conservation** - The cell cycle conservation score is a proxy for the conservation of gene program signal during data integration. It evaluates how much variance is explained by cell cycle per batch before and after integration. This should ideally be equal. Using Scanpy's `score_cell_cycle()` function we score the cell cycle stage of each cell using the gene expression data and gene sets from [46]. We then compute the variance contribution of the resulting S and G2/M phase scores using principal component regression, which is performed for each batch separately. The differences in variance before, Var_{before}, and after, Var_{after}, integration is aggregated into a final score between 0 and 1, using the equation:

   $$\text{CC conservation} = 1 - \frac{|\text{Var}_{\text{after}} - \text{Var}_{\text{before}}|}{\text{Var}_{\text{before}}}$$

   In this equation values close to 0 indicate lower conservation and 1 indicates complete conservation of the variance explained by the cell cycle. In other words, the variance remains unchanged within each batch for complete conservation, while any deviation from the pre-integration variance contribution reduces the score.
5. **Trajectory conservation** - The trajectory conservation score is a proxy for the conservation of a continuous biological signal in the joint embedding. In this metric, we compare trajectories computed after integration for relevant cell types that describe a continuous cellular differentiation process with a trajectory computed per batch and modality. Trajectories are computed using diffusion pseudotime (implemented in the `sc.tl.dpt` function in Scanpy). This approach embeds the data into a diffusion map space and computes an ordering of cells in this space from a selected root cell (a pseudotime value). As root cell, we select the cell in the earliest progenitor cluster that is most extremal in the first three diffusion components, which is still in the largest connected component of the cellular nearest neighbor graph (the graph that is used as the basis for the diffusion map computation). The conservation of the trajectory is quantified via Spearman's rank correlation coefficient s between the pseudotime

values before and after integration. The final score is scaled to a value between 0 and 1 using the equation:

$$\text{trajectory\_conservation} = (s + 1)/2.$$

Values of 1 or 0 correspond to the same order of cells on the trajectory before and after integration or the reverse order, respectively. In cases where the trajectory could not be computed, which occurs when kNN graphs of the integrated data contain many connected components, we set the value of the metric to 0. To compute a multi-modal trajectory conservation score using uni-modal ground-truth trajectories, we take the mean of the trajectory conservation scores for each modality.

A.2.2 Batch correction metrics

The following metrics assess how well an embedding removes batch variation.

1. **Batch ASW** - The average silhouette width (ASW) measures the compactness of observations with the same label in an embedding. We use the ASW to measure batch mixing by considering the non-compactness of batch labels per cell type cluster. Specifically, we consider the absolute silhouette width, $s(i)$, on batch labels per cell i. Here, 0 indicates that batches are well mixed, and any deviation from 0 indicates a batch effect. We rescale this score so that higher scores indicate better batch mixing and compute this per cell type label, j, via the equation:

$$batchASW_j = \frac{1}{|C_j|} \sum_{i \in C_j} 1 - |s(i)|$$

where C_j is the set of cells with the cell label j and $|C_j|$ denotes the number of cells in that set. To obtain the final $batchASW$ score, the label-specific $batchASW_j$ scores are averaged:

$$batchASW = \frac{1}{|M|} \sum_{j \in M} batchASW_j$$

Here, M is the set of unique cell labels. Overall, a batch ASW of 1 represents ideal batch mixing and a value of 0 indicates strongly separated batches.

2. **Graph connectivity** - The graph connectivity metric assesses whether cells of the same type from different batches are close to one another in the embedding. This is evaluated by computing a k-nearest neighbour (kNN) graph, G, on the embedding using Euclidean distances. We then check if all cells with the same cell identity label are connected on this kNN graph. For each cell identity label c, we generate the subset kNN graph $G(N_c; E_c)$, which contains only cells from a given label. Using these subset kNN graphs, we compute the graph connectivity score:

$$gc = \frac{1}{|C|} \sum_{c \in C} \frac{|LCC(G(N_c; E_c))|}{|N_c|}$$

Here, C represents the set of cell identity labels, $|LCC()|$ is the number of nodes in the largest connected component of the graph, and $|N_c|$ is the number of nodes with cell identity c. The resulting score has a range of $(0; 1]$, where 1 indicates that all cells with the same cell identity are connected in the integrated kNN graph, and the lowest possible score indicates a graph where no cell is connected.

A.2.3 Understanding variability and batch effects

To understand the extent of variability and batch effects in the benchmarking dataset, we defined train-test splits and computed the correlation of each test cell to the global or local (same cell identity) mean in training cells. Using the ATAC+GEX datasets (**Figure 4**), we find higher correlation in GEX (0.52±0.1) relative to ATAC (0.32±0.1), indicating greater variability in the ATAC data. We also observe that for both modalities, correlations are higher within donor test-train splits than across donor test-train splits, as expected, though batch effects appear larger for GEX. Within donor, imputing each cell as the mean of similarly annotated cells outperforms imputing each cell as the overall mean. However, the opposite holds when imputing across donors, another indicator of batch effects. We anticipate that successful competitors will need to take these sources or real-world donor and technical variability into account.

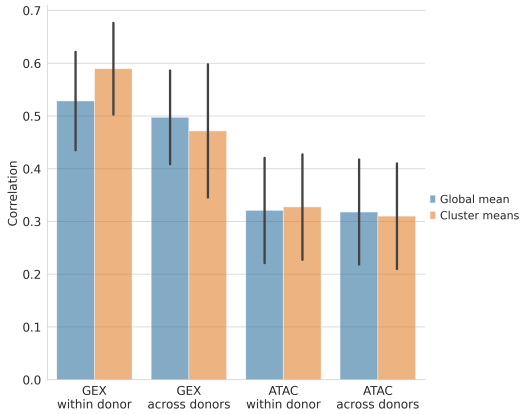

Figure 4: Comparison of within donor and between donor variability in the ATAC+GEX data. Train-test split is within or across donors. Pearson correlation is computed on log counts between each test cell and the global or local (cell identity cluster) mean of training cells. Error bars show standard deviation.

A.2.4 Metric aggregation

To rank methods, the individual metric scores will be aggregated. However, due to the differing nature of each metric, we will assign a weight to each metric after 1 month of the public competition. The goal of this weighting will be to provide equal importance on each measure when summing them. This weighting will be noted in the competition documentation and in communication to all competitions.

An overall weighted average of batch correction and bio-conservation scores will be computed via the equation:

$$S_{overall,i} = 0.6 \cdot S_{bio,i} + 0.4 \cdot S_{batch,i}$$

This reflects the relative importance of the metrics.

The batch covariate used for evaluation is "sample", however one can consider encoding the site of data collection as an additional or replacement batch covariate.

A.3 Computational benchmarking framework

The overall workflow consists of the following types. For the Data Censor, Method and Metric components, the interface specifications are task-dependent.

- **Dataset Loader**: Retrieves a dataset from a source (e.g. a HTTPS URL or S3 bucket) and store it as an AnnData file in a predetermined format.
- **Dataset Processor**: Preprocesses a dataset – for example, calculating size-factors.
- **Dataset Censor**: Separates a dataset into one or more *censored* files which will be passed to Method components, and a *solution* object which contains the ground-truth information required to evaluate a prediction.
- **Baseline Method**: A simple method for generating a prediction using the provided censored files.
- **Negative Control Method**: Serves as a negative control for the censoring and metric components. By generating constant or random predictions, negative controls should obtain bad scores on most of the metrics, unless the opposite is expected. For instance, a random embedding of a dataset will obtain a good score on any metrics which look at batch effects in the embedding.
- **Positive Control Method**: Serves as a positive control for the censoring and metric components. By returning the solution or creating a prediction based on ground-truth information, positive control methods should obtain good scores on most of the metrics, unless the opposite is expected.
- **Metric**: Calculates one or more metrics by comparing a prediction to the ground-truth solution.

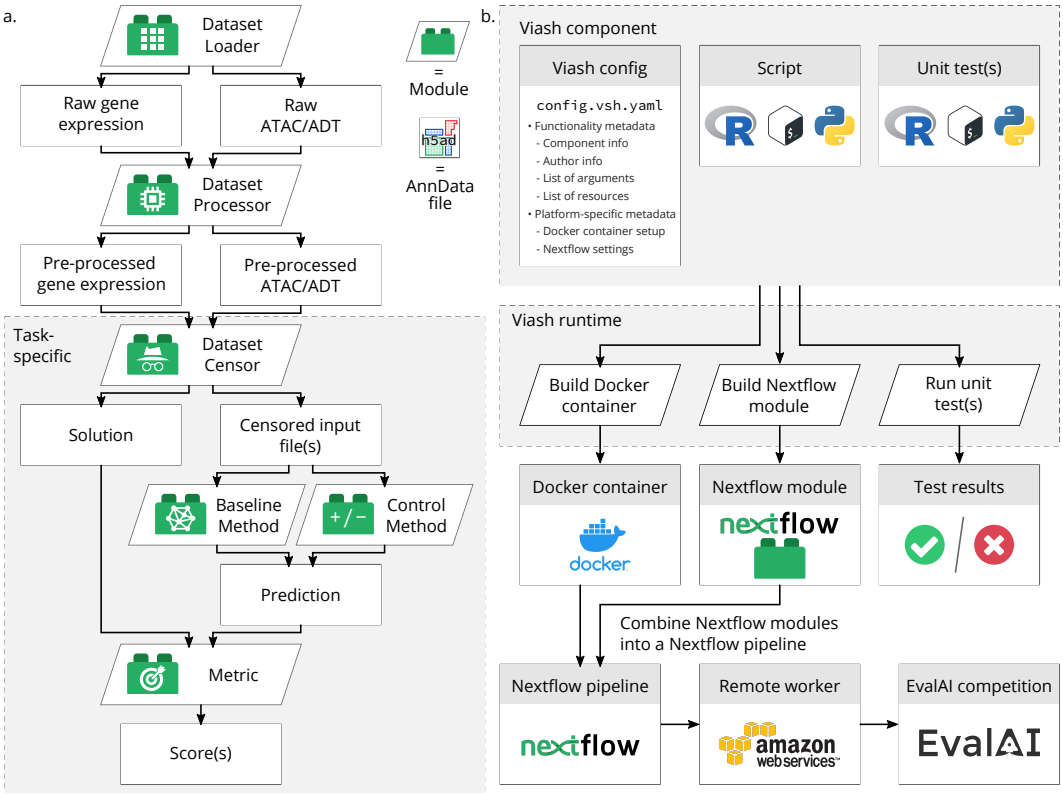

Figure 5: Overview of the computational framework. **a.** The pilot workflow consists of different types of components, for which the interfaces were defined beforehand in order to collaboratively reason about and implement new components. **b.** Several technologies were used to ensure that the pilot pipeline is reproducible (Docker), scalable (Nextflow and AWS), and versionable (Viash). Viash, in particular, was essential to be able to rapidly prototype new components in a collaborative and time-constrained setting.

The benchmarking framework was developed with a combination of technologies to allow for reproducibility and scalability without sacrificing on ease of use and rapid prototyping (Figure 5B). These are the following:

- **AnnData**: All datasets are formatted using the Annotated Data file format from the Scanpy framework [38]. AnnData is a lightweight but efficient format, which requires minimal package dependencies in order to load a dataset into memory. Python and R users can use the `anndata` package on PyPi and CRAN, respectively.
- **Viash** [40]: A tool for wrapping small scripts and some metadata as modular pipeline components. Examples of such metadata include author information, a list or arguments required by the script, or a list of R and Python packages which the component requires. Viash can be used to perform a variety of tasks, including wrapping the component as a standalone Bash executable or Nextflow module and unit testing the component.
- **Docker**: Each component has a corresponding (implicit) Docker container. When a version of the benchmark pipeline is released, the containers are pushed to Docker Hub to ensure reproducibility on many systems.
- **Nextflow** [47]: One of the more popular frameworks for defining and running pipelines in Bioinformatics. By having extensive support for containerisation of components and interfacing with cloud execution and storage solutions, Nextflow allows for flexibility in switching our chosen cloud solution for alternatives if so desired.
- **EvalAI** [48]: An open-source framework for evaluating machine learning algorithms at scale. Through the EvalAI infrastructure, competitors can submit solutions. This triggers a remote

evaluation worker hosted on AWS EC2 which executes a Nextflow evaluation pipeline on the user-submitted files. After the evaluation pipeline has finished running, a competitor can browse through the overall ranking of methods, including baseline results generated by this benchmarking framework.

Since the components included in this benchmarking framework were developed collaboratively, a major benefit of using Viash to generate Docker containers and Nextflow modules is that it allows for separation of concerns. By separating the pipeline logic from the core functionality provided in each of the components (written as R or Python scripts), component developers did not directly need to interface with the Nextflow Domain Specific Language (DSL), which can form a steep barrier to entry for novice pipeline component developers.

A.4 Challenges and logistics associated with building a multimodal single-cell sandbox

Building this sandbox for multimodal single-cell data required coordinating technical expertise across the US and Europe. This required management of data generation, data analysis, and designing computational infrastructure. The following section describes the challenges and key learnings associated with each of these arms of the initiative.

A.4.1 Data Generation

Data generation was the most challenging aspect of the initiative to organize. Generating single-cell data is not easy and requires separate PhD-level scientists to write the protocols, isolate cells, prepare the single-cell libraries, and operate the sequencing machines.

Sourcing reagents One of the biggest hurdles we faced was delays in the supply chain due to COVID-19. When we initially contacted vendors to source bone-marrow mononuclear cells, we found only one had enough inventory in May 2021 to support data generation across sites. We then faced customs delays shipping cells from the vendor in California, USA to Munich, Germany. We faced a similar hurdle sourcing the antibody panels for the CITE-seq data generation. When we contacted the vendor in May 2021, we were told the stock panels were backordered through August 2021. To get antibody panels in time, we arranged access to a pre-market product that is now commercially available. In July 2021, we hit a shortage of sequencing reagents that affected all sites and delayed sequencing of constructed libraries. These issues were compounded by the just-in-time inventory stocking policies at partners. Throughout the competition, we learned to keep extra inventory on hand to account for unplanned repeat experiments, which ended up being more common than we anticipated.

Difficulty in sample preparation Like most humans, we fell prey to the planning fallacy [49] and anticipated that sample preparation would be straightforward and work correctly on the first try. Instead, we faced difficulties at every stage of data generation. Only two of the four sites had generated both GEX+ATAC and GEX+ADT libraries. None of the participants at the sites had experience with bone marrow mononuclear cells. Start to finish generating a single dataset takes no shorter than three weeks, with little feedback about the experiment success along the way. Three sites experienced challenges with cell isolation that led to a two month delay in preparing data due to a need to repeat experiments. Two sites faced unexpected failures in sequencing that led to a month delay in sequencing libraries. We found interestingly enough that the success of each site was unrelated to running pilot experiments, however data at each site did seem to get better with repeats.

Project management Making sure scientists involved with data generation knew what to do when was critical to the success of the project. Through this process, we learned to adopt project management best practices like using centralized documents to track the status of each sample at each site, list all protocols, outline clear timelines, and keep track of the accountability for each site. We organized weekly planning meetings to review the project timeline and discuss our experimental plans and results.

A.4.2 Data Analysis

Prior to this effort, none of the sites had experience analyzing multimodal single-cell data in human bone-marrow mononuclear cells. Devising analysis strategies required consulting existing literature

and contacting domain experts. We then needed to create template analysis notebooks and train a team of 7 data analysts to perform the analysis. This process required constant supervision and iteration to revise cluster labels and ensure data quality. We set up a system where two of the organizers picked one data type each to review. Analysis would then work on QC, initial annotation, and doublet identification for a dataset, submit their work to the relevant reviewer, and incorporate feedback over the course of a week per dataset. This attention to detail in the data analysis was crucial to removing doublets and low quality cells, which compromise dataset quality.

A.4.3 Computational Infrastructure

Portable submission components One of the biggest challenges with this competition was designing infrastructure that enabled competitors to submit runnable code in a variety of languages on a centralized data server and as part of a workflow. We also wanted to accommodate participants who may have more experience scripting than creating portable docker containers compatible with our testing infrastructure. To achieve these goals, we used Viash, a tool to create portable command line interfaces using a script and a configuration YAML. The details of this infrastructure is described above.

Documentation Making this competition accessible necessitated documenting the computational infrastructure and data. Although Viash solves many of our difficulties of centralized benchmarking, most users are unfamiliar with it. Additionally, single-cell data is not a common substrate for machine learning tasks. Many NeurIPS attendees may not be familiar with the data type, and this may provide a barrier for entry. Finally, the tasks presented in this sandbox were mostly formulated from scratch. To make sure this sandbox is accessible, we made sure to fully document every aspect including quickstart guides and walkthroughs of the development process.

We also knew that even with the documentation, participants would have more detailed questions. To encourage a public discourse around areas of confusion, we set up a Discord server where anyone could ask questions. So far this has been a successful venue for handling both technical questions and hosting public discussion around the tasks with over 400 members.

Appendix References

[41] Aaron T. L. Lun, Karsten Bach, and John C. Marioni. Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biology*, 17(1):75, dec 2016.

[42] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29):861, sep 2018.

[43] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, dec 2019.

[44] Haley M Amemiya, Anshul Kundaje, and Alan P Boyle. The ENCODE Blacklist: Identification of Problematic Regions of the Genome. *Scientific reports*, 9(1):9354, jun 2019.

[45] Darren A Cusanovich, Riza Daza, Andrew Adey, Hannah A Pliner, Lena Christiansen, Kevin L Gunderson, Frank J Steemers, Cole Trapnell, and Jay Shendure. Multiplex single cell profiling of chromatin accessibility by combinatorial cellular indexing. *Science (New York, N.Y.)*, 348(6237):910–914, may 2015.

[46] Itay Tirosh, Benjamin Izar, Sanjay M. Prakadan, Marc H. Wadsworth, Daniel Treacy, John J. Trombetta, Asaf Rotem, Christopher Rodman, Christine Lian, George Murphy, Mohammad Fallahi-Sichani, Ken Dutton-Regester, Jia-Ren Lin, Ofir Cohen, Parin Shah, Diana Lu, Alex S. Genshaft, Travis K. Hughes, Carly G. K. Ziegler, Samuel W. Kazer, Aleth Gaillard, Kellie E. Kolb, Alexandra-Chloé Villani, Cory M. Johannessen, Aleksandr Y. Andreev, Eliezer M. Van Allen, Monica Bertagnolli, Peter K. Sorger, Ryan J. Sullivan, Keith T. Flaherty, Dennie T. Frederick, Judit Jané-Valbuena, Charles H. Yoon, Orit Rozenblatt-Rosen, Alex K. Shalek, Aviv Regev, and Levi A. Garraway. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science*, 352(6282):189–196, Apr 2016.

[47] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.

[48] Deshraj Yadav, Rishabh Jain, Harsh Agrawal, Prithvijit Chattopadhyay, Taranjeet Singh, Akash Jain, Shiv Baran Singh, Stefan Lee, and Dhruv Batra. Evalai: Towards better evaluation systems for ai agents. *arXiv preprint arXiv:1902.03570*, arXiv:1902.03570, 2019.

[49] Roger Buehler, Dale Griffin, and Michael Ross. Exploring the "planning fallacy": Why people underestimate their task completion times. *Journal of personality and social psychology*, 67(3):366, 1994.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] , See associated Datasheet **Section 1.I**
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] , See associated Datasheet **Section 7**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] , See associated Datasheet **Section 7**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] , See associated Datasheet **Section 4.F** and **Section 5.A**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] , See **Section 3.1**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] , see Datasheet **Section 5.B**
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] , see Datasheet **Section 5**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] , see Datasheet **Section 7**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] , see Datasheet **Section 7**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] , see Datasheet **Section 7**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] , see Datasheet **Section 7**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No] , compensation was arranged by AllCells.