PsychENCODE snRNA-seq uniform processing pipeline

Summary
The snRNA-seq processing pipeline was constructed on the basis of published best practices, as well as based on new benchmarking metrics for existing methods. The pipeline has been implemented in Python for the most part, except for the final cell type annotation steps which involved the R-based program Azimuth. The overall workflow can be summarized in three main steps:

1. Count matrix generation, demultiplexing and ambient RNA clean-up
2. Per-fastq set/sample processing using Pegasus[1,2]
3. Per-study aggregation of processed sample and cell-type annotation

In the following, we present each of the steps in greater detail.

1. Count matrix generation, demultiplexing and ambient RNA clean-up

A schematic for the workflow is presented in Figure 1.

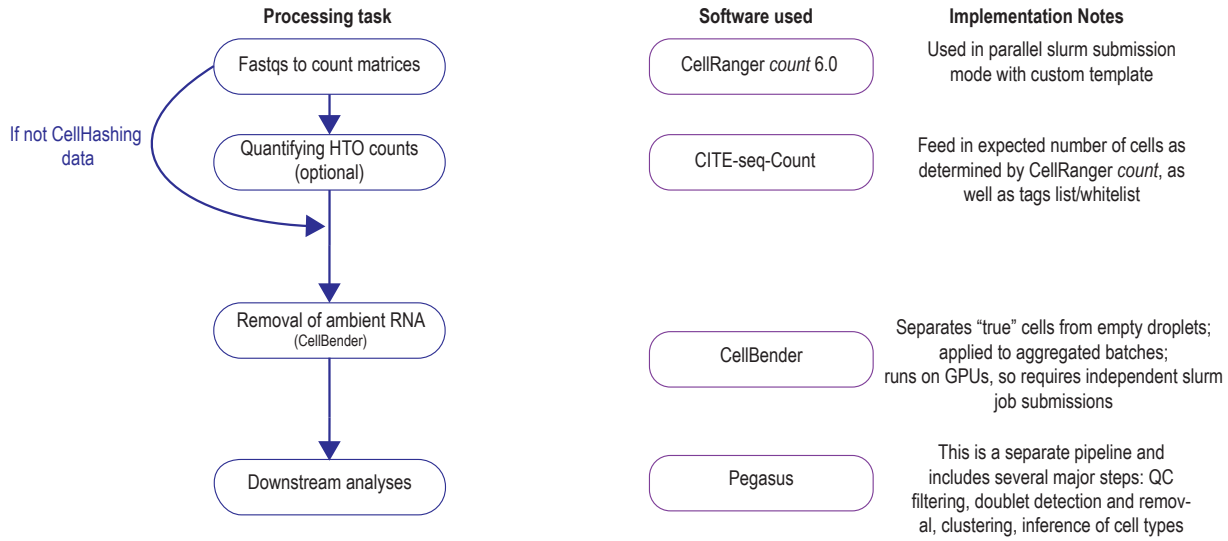

Figure 1. *Schematic for Part 1 of the workflow.*

Count matrix generation: The count matrix is generated using CellRanger *count* v6.0[3]. Each sample was run independently and no aggregation (using CellRanger *aggr*) was ever carried out. However, if the same sequencing sample was run through multiple lanes, and thus shared the same sequencing sample identifier with different lane numbers (L001, L002, etc.), we used the CellRanger *count* feature of automatically pooling all the contents of the 'same-sample different-lanes' fastqs. We also used the '--include-introns' flag for all the samples as the studies under consideration all involve single-nucleus RNA-seq, and therefore necessitate a quantification of the pre-mRNA (and thus pre-spliced) transcripts as well. Additionally, we specified the chemistry of the sample being quantified, whether based on 10x genomics v2 or v3 chemistry. Finally, the same initial procedure for per-cell RNA quantification was followed

for the MULTI-seq and CellHashing data as well, with the demultiplex step using the corresponding HTO files described in the following.

Demultiplexing: The studies analyzed included data from MULTI-seq and CellHashing multiplexing assays. To process this data, we first needed to quantify the per-cell HTO counts. There are two standard programmatic options for this: using the R-based deMULTIplex package[4,5] and the command-line based CITE-seq-Count package[6,7]. For reasons of programmatic convenience, we chose to use the CITE-seq-Count package (v1.4.x), with the command embedded into out Python scripts. The options to be provided for this command include: the R1 and R2 fastq files for the HTO reads; a csv file containing the HTO barcode and an associated name for the hashtag to be included in downstream HTO quantifications; the first (*cbf*) and last (*cbl*) locations for the cell barcodes in the file; the first (*umif*) and last (*umil*) locations for the UMIs in the file, which are chemistry-specific; and the approximate number of cells expected in the sample or a list of cell barcodes. For the tags file, we used the data providers' lists of HTO barcodes followed by generic names – Hashtag_1, Hashtag_2, etc. We set *cbf* = 1, *cbl* = 16, *umif* = 17, and *umil* = 26 for v2 chemistry samples and *umil* = 28 for v3 chemistry samples. We chose to provide an expected number of cells instead of explicit cell barcodes. We supplied the expected number of cells using the *metrics_summary.csv* file output by CellRanger. From that file, the number of cells output by CellRanger's own cell-identification algorithm was extracted and 500 was added to it as a simple buffer (in case CellRanger's algorithm undercounted cells a little).

Ambient RNA clean-up: To more carefully separate out true cells from empty droplets with ambient RNA, we used the program *remove-background*[8] from the *CellBender* package[9]. We used the program in command-line form wrapped in our Python script. Specifically, the program is optimized to run on GPUs (there is a significant difference in the runtime depending on whether GPUs or CPUs are used), and accordingly we implemented this portion of the pipeline on a GPU (adding the--cuda flag). The input to the program is the raw (i.e. without filtering for cells identified by CellRanger) output .h5 file from CellRanger *count*. The options included in the program run are: a target false positive rate (--fpr) of 0.01; the number of training epochs (--epochs) = 150; a rough expected number of cells (--expected-cells) = the output in *metrics_summary.csv* from CellRanger *count*, as in the CITE-seq-Count run; the total number of droplets to be included (--total-droplets-included) in the analysis, set to be the expected number of cells + 20,000 (chosen to be large enough to encompass many empty droplets for the training). The output of this step is a .h5 file, where the empty droplets are filtered out, leaving just the inferred true cells. This .h5 file is chosen as the input for the downstream analyses in Pegasus.

2. Per-fastq set/sample processing using Pegasus

The primary steps for this part of the analysis are outlined in Figure 2. The steps are applied to the CellBender output for each sample separately (though some pooling may have been carried out across parallel lanes for the same sample, as described above). Many of the steps are applied in parallel to those in the Pegasus tutorial[1]. After filtering cells based on the lower bounds shown in Figure 2, we removed all genes included in the MitoCarta v3.0 database[10], such as mitochondrial genes and certain genes highly correlated with RNA sample quality (see,

for example, Hodge et al 2019[11]).

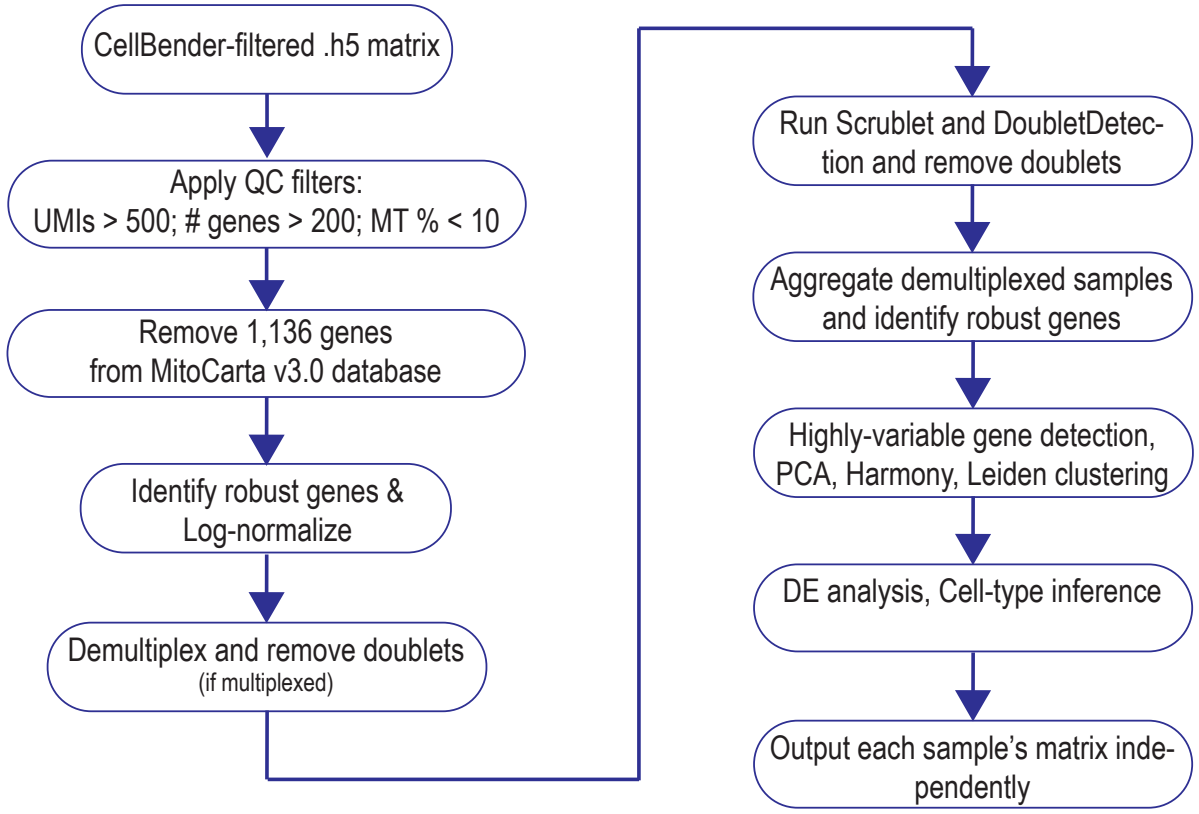

Figure 2. *Schematic for Part 2 of the analysis in Pegasus.*

The robust genes were identified and the counts matrix was log-normalized using the default options in Pegasus.

At this stage, if the data is from a multiplexed experiment, the matrices are decomposed into cells from each of the component samples using the Pegasus *demultiplex* algorithm. The input to this are the matrix, feature and barcode files from the CITE-seq-Count run. Only cells that are identified as singlets in the demultiplexing step are retained.

Next, doublets are identified using a combination of two computational methods. These steps are carried out for the demultiplexed samples as well. The rationale is that demultiplexing removes inter-sample doublets, while intra-doublet samples still need to be removed. We found that a combination of the program Scrublet[12] in default mode and DoubletDetection[13] worked well. The parameters for the DoubletDetection *BoostClassifier* algorithm include: *n_iters* = 25, *use_phenograph* = *False*, *standard_scaling* = *True*. The subsequent *predict* function employs the parameters *p_thresh* = 1 e-16, and *voter_thresh* = 0.3.

After the doublet removal, we aggregate the demultiplexed samples again for robust gene identification, highly variable gene selection (5000 genes chosen), PCA, batch correction using Harmony[14], nearest-neighbor detection, Leiden clustering and UMAP dimensionality reduction.

We carry out differential expression analysis using the t-test, where the expression of genes in every cluster is compared against all others. Finally, cell type inference is carried out based on a series of marker gene collections using the *infer_cell_types* function. It should be noted that many of these last steps (from the highly variable gene selection onwards) are not strictly necessary for further analysis. The raw count matrices for each sample, after doublet removal and demultiplexing, are used in the next steps without referencing any of the cell-type annotations in this per-sample stage.

3. Per-study aggregation of processed sample and cell-type annotation

The schematic depicting the set of steps for this stage of the workflow is shown in Figure 3.

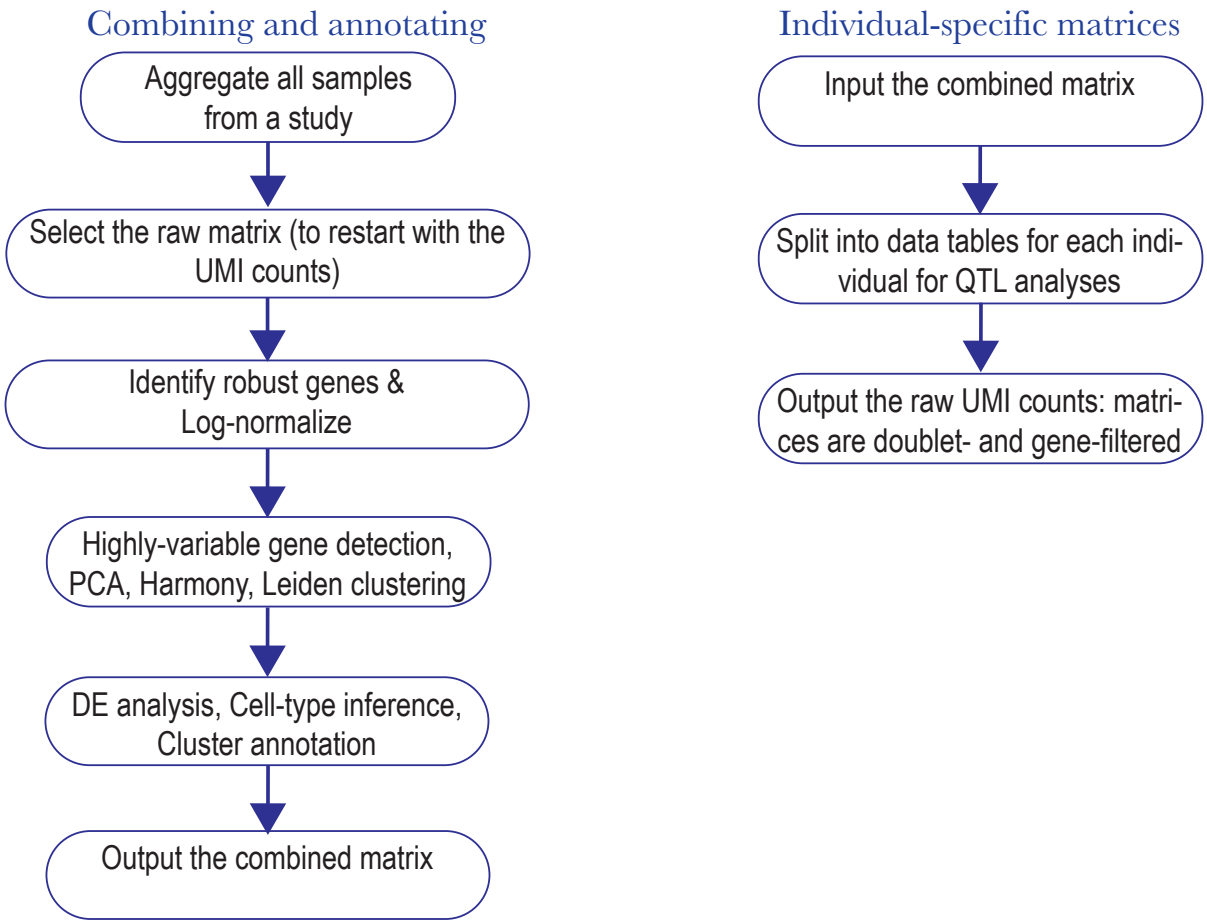

Figure 3. *Schematic for Stage 3 of the analysis. This is the per-study analysis.*

First, we aggregate all samples across each study, with the demultiplexed sample names being used as identifiers for the MULTI-seq and CellHashing datasets. We specifically select the raw count matrices in the following analyses. We then proceed with the joint analysis steps on the aggregated AnnData object in Pegasus. Many of the steps remain the same as above (no

doublet detection is carried out at this stage), up until the cluster annotation process. One difference is that we increase the Leiden clustering resolution parameter between 4.0 and 6.0. The intention is to recover as many cells as possible by more finely dividing the clusters, before removing those clusters that failed to be annotated (as described in the following). The number varied simply because the limit for UMAP generation was 64 clusters, and the same resolution parameter resulted in different numbers of clusters for different studies.

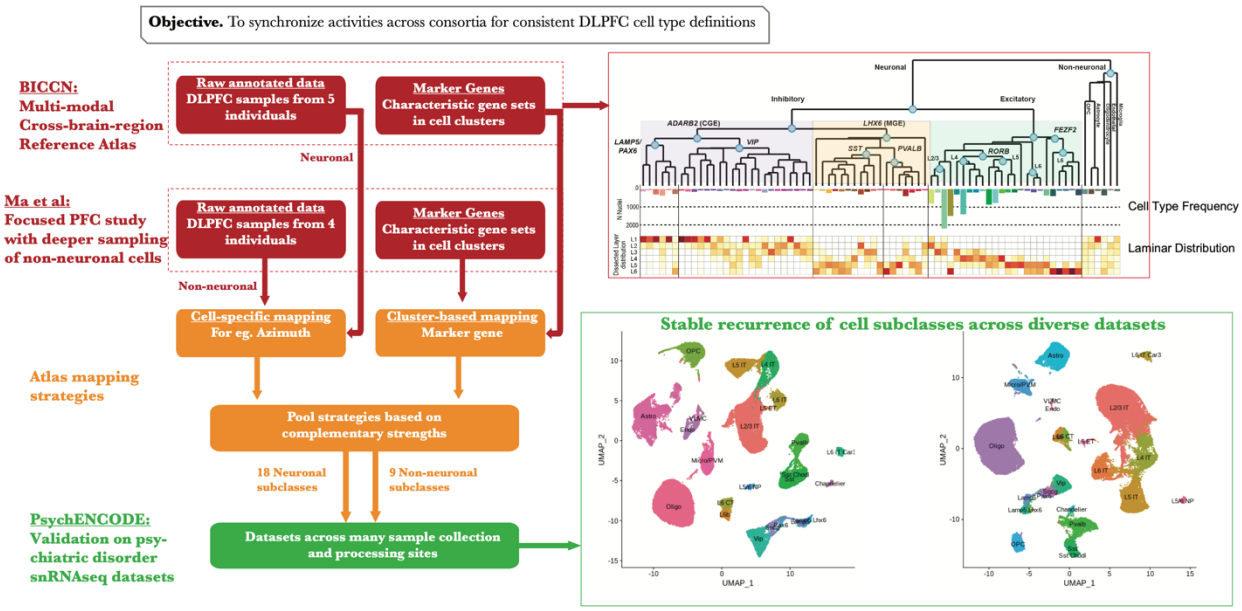

Figure 4. *Outline of methods used to generate the hybrid cell annotation scheme, merging neuronal types from the BICCN and the non-neuronal types from Ma et al.*[15]

Cluster annotation: Cluster annotation proceeds in a two-step process (**Figure 4**). We first use Pegasus' *infer_cell_types* function to associate the Leiden clusters with reference cell types based on the hybrid marker gene sets obtained from merging the neuronal subclass markers from the BICCN and the non-neuronal subclasses from Ma et al.[15] (*Documentation_merged_subclass_markers.json*). The subclass annotations include the following:

> **Excitatory Neurons:** *L2/3 IT*, *L6 IT*, *L4 IT*, *L5 IT*, *L6 IT Car3*, *L5 ET*, *L6 CT*, *L5/6 NP*, *L6b* (L# signifies the cortical layer context; IT = intra-telencephalic, ET = extra-telencephalic, CT = Cortico-thalamic, NP = Near-projecting)
> **Inhibitory Neurons:** *Lamp5*, *Pax6*, *Sncg*, *Vip*, *Lamp5 Lhx6*, *Chandelier*, *Pvalb*, *Sst*, *Sst Chodl*
> **Non-neuronal cells:** *Astro* (Astrocytes), *Endo* (Endothelial cells), *VLMC* (Vascular Leptomeningeal cells), *Micro* (Microglia), *Oligo* (Oligodendrocytes), *OPC* (Oligodendrocyte precursor cells), *Immune*, *RB* (*Red Blood lineage cells*), PC (*Pericytes*), SMC (*Smooth Muscle Cells*).

These annotations, in turn, were used at this stage of the analysis because of their purported robustness across brain regions and species (see, for example, the discussion in Bakken et al

2020[16]). Given these annotations, and the most likely assignments (sometimes a many-to-one assignment of subclasses to clusters), we used Pegasus' *infer_cluster_names* and *annotate* functions to assign a single best-fit subclass assignment. We then removed all unassigned clusters. Our rationale in doing so was that these unassigned clusters mainly consisted of low-expression cells, likely reflecting cellular debris. For example, we found that some clusters were enriched for synaptic and transmembrane protein expression, while the overall number of genes expressed was very small. We speculated that these clusters included extracellular debris and accordingly removed them. We thus chose a conservative approach to cluster inclusion in our downstream analyses.

However, we did not finalize the cell assignments based on the marker gene analysis, because of the ambiguity of the resolution in cluster-based assignment strategies. Instead, the marker gene analysis was used for the aforementioned cluster removal. The remaining clusters were then processed through the Seurat/Azimuth[17] pipelines to assign the subclass annotations. The advantage of the Azimuth approach is that it uses cell-based assignment: that is, each cell is individually assessed against a reference cell atlas to find the most likely assignment of cell type. Specifically, the .h5ad objects from the Pegasus processing (after the aforementioned cluster removal) were read into Seurat objects using functions in the *SeuratDisk* package, and processed: first, convert the AnnData object into a .h5Seurat object; second, load in the .h5Seurat object using the *LoadH5SeuratObject* function; third, due to issues with reading in the raw counts through this mechanism, the raw counts for each study were independently exported into .npz format and reassigned to the Seurat object in the "counts" slot; fourth, the dataset was processed using the *SCTransform*[18] function in Seurat; finally, the *FindTransferAnchors* and *TransferData* functions were used to add the subclass assignments as metadata in the Seurat object. We exported these assignments to a separate file for integration with the Pegasus objects downstream. To transfer anchors we performed the same preprocessing steps on the raw counts for the reference atlas cells from the BICCN (118,291 cells pre-annotated with the reference neuronal subclasses) and the Ma et al study (172,120 cells pre-annotated with the reference subclasses). Note that the marker genes for the subclasses were derived from the same dataset.

Merging of the results from the two reference atlases. We first annotated all the cells with the BICCN and Ma et al. schemes separately. To reconcile the results, we performed the following steps:

1. Look at the cell type label for each annotation scheme, and assign cell types to the appropriate cell class, i.e. whether that cell type is Excitatory, Inhibitory, or Glial.
2. If the cell classes are same for the BICCN and the Ma et al. schemes, keep the cell barcode. If the cell classes are different, remove the cell barcode. We have found that cells that are mismatches in cell class between the two schemes tend to be of lower technical quality, measured in terms of the number of genes expressed and number of UMI counts.

3. For all the cell barcodes that are retained, if the cell class is Excitatory or Inhibitory keep the BICCN label in the final annotation; if the cell class is Glial, keep the Ma et al. annotation.

4. These new annotations were merged with the h5ad objects, by filtering out the cells which could not be reconciled at the cell-class level.

Once the subclasses were assigned, we added them to the metadata in the Pegasus .h5ad (AnnData) objects. The .h5ad objects were rerun through dimensionality reduction, and reclustered so that the UMAP coordinates reflected the post-filtration groups of cells. We provide these annotated objects for download. The matrix "raw.X" contains the raw UMI counts, and "X" contains the log-normalized counts.

The annotations of the .obs dataframe of the AnnData object include marker gene annotations ("anno") of clusters as well as the Azimuth labels ("azimuth", "subclass"). **The final annotations are contained under the "subclass" column.**

We also provided the individual IDs associated with each cell and sample. This is especially relevant for the multiplexed datasets. In addition to the .h5ad files, we have also uploaded tab-delimited files for each **individual** in the analyses, where the genes (with HGNC symbols NOT Ensembl IDs) mark the rows and the cells are arrayed along the columns. The column headers are the Azimuth labels of the cells. We emphasize the fact that each individual was assigned a separate matrix: this means that if parallel samples were sequenced for the same individual, the final matrix pools all the cells from these samples.

(Documentation prepared by Prashant S. Emani, prashant.emani@yale.edu)

References

1. Li, B. et al. Pegasus for Single Cell Analysis. Available at: https://pegasus.readthedocs.io/en/stable/.
2. Li, B. *et al.* Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq. *Nat. Methods* **17**, 793–798 (2020).
3. 10x Genomics. Single Library Analysis with cellranger count.
4. McGinnis, C. deMULTIplex package.
5. McGinnis, C. S. *et al.* MULTI-seq: sample multiplexing for single-cell RNA sequencing using lipid-tagged indices. *Nat. Methods* **16**, 619–626 (2019).
6. Stoeckius, M. *et al.* Cell Hashing with barcoded antibodies enables multiplexing and doublet detection for single cell genomics. *Genome Biol.* **19**, 224 (2018).
7. CITE-seq-Count. doi:https://zenodo.org/record/2590196
8. Fleming, S. J., Marioni, J. C. & Mehrtash, B. CellBender remove-background. Available at: https://cellbender.readthedocs.io/en/latest/citation/index.html.
9. Fleming, S. J., Marioni, J. C. & Babadi, M. CellBender remove-background: a deep generative model for unsupervised removal of background noise from scRNA-seq

datasets. *bioRxiv* (2019). doi:10.1101/791699
10. Rath, S. *et al.* MitoCarta3.0: an updated mitochondrial proteome now with sub-organelle localization and pathway annotations. *Nucleic Acids Res.* **49**, D1541–D1547 (2021).
11. Hodge, R. D. *et al.* Conserved cell types with divergent features in human versus mouse cortex. *Nature* **573**, 61–68 (2019).
12. Wolock, S. L., Lopez, R. & Klein, A. M. Scrublet: Computational Identification of Cell Doublets in Single-Cell Transcriptomic Data. *Cell Syst.* **8**, 281-291.e9 (2019).
13. Gayoso, A., Shor, J., Carr, A. J., Sharma, R. & Pe'er, D. DoubletDetection. doi:http://doi.org/10.5281/zenodo.2678041
14. Korsunsky, I. *et al.* Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat. Methods* **16**, 1289–1296 (2019).
15. Ma, S. *et al.* Molecular and cellular evolution of the primate dorsolateral prefrontal cortex. *Science* **377**, eabo7257 (2022).
16. Bakken, T. E. *et al.* Evolution of cellular diversity in primary motor cortex of human, marmoset monkey, and mouse. *bioRxiv* (2020). doi:10.1101/2020.03.31.016972
17. Hao, Y. *et al.* Integrated analysis of multimodal single-cell data. *Cell* **184**, 3573-3587.e29 (2021).
18. Hafemeister, C. & Satija, R. Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biol.* **20**, 296 (2019).