SUPPLEMENTARY MATERIALS

Contents

1 Datasets and pre-processing

1.1 Dataset Descriptions

Dataset	Distribution	Cells	K	Description
10x_pooled_full	Poisson	73233	8	This dataset consists of 8 FACS sorted cell types from Zheng *et al.* (2017): CD19+ b cells, CD14+ monocytes, CD34+, CD56+ NK, CD4+/CD45RO+ memory t, CD8+/CD45RA+ naive cytotoxic, CD4+/CD45RA+/CD25- naive t, and CD4+/CD25 regulatory t
10x_PBMC	Poisson	68579	10	68k PBMC dataset from Zheng *et al.* (2017), ground truth labels inferred by Spearman correlation with mean expression values of cell-sorted data
Zeisel	Poisson	3005	9	Zeisel *et al.* (2015)
Zeisel_sub	Poisson	3005	9	Zeisel *et al.* (2015), with UMI counts downsampled by 99%
10x_1.3M	Poisson	1306127	10*	1.3 million cell dataset from 10x Genomics
Tasic	Poisson	1629	49	Tasic *et al.* (2016) - cells from adult mouse visual cortex
mESC	LogNorm	182	3	Buettner *et al.* (2015), used as tests dataset in Wang *et al.* (2017)
Kolod	LogNorm	704	3	Kolodziejczyk *et al.* (2015), used as test dataset in Wang *et al.* (2017)
Pollen	LogNorm	249	11	Pollen *et al.* (2014), used as test dataset in Wang *et al.* (2017)
Usoskin	Poisson	622	4	Usoskin *et al.* (2015), used as test dataset in Wang *et al.* (2017)

Table 1: Dataset descriptions. Note: the 10x_1.3M dataset does not have a provided K. We chose K=10 as a balance between runtime performance and cluster resolution. All the 10x datasets were downloaded from https://support.10xgenomics.com/single-cell-gene-expression/datasets.

1.2 Dataset Pre-processing

All methods were run on the same subset of genes, selected by first binning the genes into five bins by mean expression value, and then taking the top 20% of genes in each bin by variance. For most datasets, the NMI did not change substantially for any of the methods when larger sets of genes were used.

1.3 Computational Experiment Details

tSNE used the implementation in scikit-learn 0.19. tSNE and ZIFA used 2 output dimensions. tSNE was run after first taking the truncated SVD with 50 dimensions on the log-transformed and column-normalized data, using the scikit-learn randomized TSVD. SIMLR, Magic, and UNCURL were all run using default settings. Large-scale SIMLR (Python implementation) was used for all datasets with more than 1000 cells, with 500 PCA components, on log-transformed data. UNCURL was run using 8 threads for all datasets with less than 1000 cells and 32 threads for all other datasets.

All computational experiments were done on a cluster with 64 dual-core AMD Opteron 6380 processors and 512GB of memory.

2 Additional Algorithmic Details

2.1 State estimation

Algorithm 1 State estimation with the Probabilistic Convex Mixture Model

function ESTIMATE-STATE($X, k, maxiters, \epsilon$)
  $W \leftarrow$ Init-W(X, k)
  $M \leftarrow$ Init-M(X, W, k)
  for $iter \leftarrow 1...maxiters$ **do**
    Update W to minimize $-\log P(X \mid M, W)$, subject to W nonnegative
    Update M to minimize $-\log P(X \mid M, W)$, subject to M nonnegative
    if M and W changed less than ϵ this iteration **then**
      return M, W normalized
    end if
  end for
  return M, W normalized
end function

By default, Init-W(X, k) simply performs a k-means clustering on D reduced with truncated SVD, and assigns 0.75 to the highest cluster for each cell and 0.25 divided among the remaining clusters. Init-M(X, W, k) sets each column of M to be the mean of all cells assigned to that cluster.

W and M can also be initialized manually, or with the output of qualNorm or a different clustering/dimensionality reduction algorithm.

2.2 Sparse NoLips Algorithm

Algorithm 2 One round of NoLips optimization for W

function NoLips-Estimate-W(X, M, W, k, ϵ)
  for $c \leftarrow 1...numcells$ **do**
    $\lambda \leftarrow 1/(2\sum_j X[j,c])$
    $ci \leftarrow [0]^k$
    for $g \leftarrow 1...numgenes$ **do**
      $mw \leftarrow X[g,c]/(M[g,:] * W[:,c])$
      for $j \leftarrow 1...k$ **do**
        $ci[j] \leftarrow ci[j] + mwM[g,j]$
      end for
    end for
    for $j \leftarrow 1...k$ **do**
      $W \leftarrow \max(0, W[j,c]/(1 + \lambda W[j,c](\sum_l M[l,j] - ci[j])))$
    end for
  end for
  return W
end function

The optimization step for M proceeds identically, with X^T, W^T, and M^T as the parameters to the above algorithm.

2.3 Identifying and ranking cluster specific genes

In Figure 6 of the main text, we utilized the ranked list of cluster specific genes. While a commonly used approach used for cluster specific gene identification is 'one-vs-all' differential gene expression analysis, this method suffers from a few defects making it unsuitable for identification of over-expressed cell type specific genes from multiple clusters. The main problem with the one-vs-all approach is that it treats all other clusters equally, when focusing on one cluster. This can lead to overlap among significant gene sets between different clusters. Moreover, differential gene expression analysis is also sampling model dependent hence would not work well for our pre-processed data. Hence, we devise the following test statistic to compare how over-expressed a gene is on average in one cluster compared to all other clusters:

$$CScore[G,C] = \frac{E[X_{j,i\in S_C}] + \epsilon}{\max_{k'\in[K],k'\neq C} E[X_{j,i\in S_{k'}}] + \epsilon}$$

Here G is the gene, C is the cluster of interest, S_k is a set of all cells belonging to cluster k and ϵ is a user defined small number ($\epsilon = 10^{-3}$ was used in this paper). The test statistic measures the ratio of average expression between the cluster of interest and the highest among all other clusters. Hence, if $CScore \geq 1$, it represents a gene that has a higher expression in the cluster of interest than all other clusters. Since the statistic basically measures fold change, it can be used to rank the cluster specific genes.

2.4 Heatmap of cluster specific gene expression

Clusters are identified by performing $arg - max$ on the W matrix estimated by UNCURL. The top 10 genes of each cluster are identified by sorting on the basic of the $CScore$. We then group the genes and cells by cluster. Finally, within each cluster (say cluster i), the cells are sorted based on the values of the ith column of W. Each gene's expression is then normalized by it's maximum value. A heatmap is then plotted on this new matrix.

3 Synthetic data generation for lineage inference

3.1 Generating simulation data along a linear trajectory

To generate sampled data along a linear trajectory, we first select the extreme means M_1 and M_2. We then generate the true state matrix T such that the ith column is equal to:

$$T_i = rM_1 + (1 - r)M_2$$

Where $r \in [0, 1]$ is a variable that controls the mixing between the two means. The r's are chosen sequentially to obtain a complete lineage between the two simulated cell types. The observed data matrix (D) is then generated by the following way:

$$D_{i,j} = r, \text{where } r \sim Poiss(T_{i,j})$$

The normalization of the read count (to yield a standard read depth for each cell) is then performed using the uniform sampling method described in (Heimberg *et al.*, 2016).

3.2 Generating simulation data along a branched trajectory

To generate sampled data along a branched trajectory, we first select the three extreme means M_1, M_2 and M_3. The centroid M_c is then found as follows:

$$M_c = \frac{M_1 + M_2 + M_3}{3}$$

Having generated this, we then proceed to generate linear trajectories between the pairs (M_1, M_c), (M_2, M_c) and (M_3, M_c). There are then combined into a true state matrix T. We then generate the observed data matrix in the same way as the linear case.

4 Additional results

4.1 Clustering NMI results

Dataset	tSNE + kmeans	SIMLR + kmeans	ZIFA + kmeans	Magic + tSNE + kmeans	UNCURL+ argmax	UNCURL+ tSNE + kmeans
10x_pooled_full	0.65	0.59	N/A	0.54	**0.83**	0.56
10x_PBMC	0.42	0.42	N/A	0.31	**0.57**	0.38
Zeisel	0.72	**0.77**	0.42	0.53	0.73	0.67
Zeisel_sub	0.55	0.58	0.43	0.37	**0.64**	0.61
Tasic	0.78	0.71	0.48	0.64	**0.82**	0.79
mESC	0.38	**0.81**	0.41	0.3	0.51	0.54
Kolod	0.99	0.99	0.55	**1.0**	0.95	**1.0**
Pollen	**0.95**	**0.95**	0.81	0.88	**0.95**	**0.95**
Usoskin	0.79	0.74	0.58	0.79	0.87	**0.98**

Dataset	kmeans	PCA + kmeans	Magic + kmeans	Magic + PCA + kmeans	UNCURL+ kmeans	UNCURL+ PCA + kmeans
10x_pooled_full	0.34	0.59	0.71	0.71	**0.85**	0.7
10x_PBMC	0.4	0.25	0.07	0.07	**0.57**	0.5
Zeisel	0.59	0.44	0.49	0.48	0.74	0.55
Zeisel_sub	0.43	0.43	0.42	0.37	**0.68**	0.5
Tasic	0.72	0.6	0.64	0.64	0.81	0.73
mESC	0.4	0.32	0.29	0.3	0.62	0.56
Kolod	0.92	0.65	**1.0**	**1.0**	0.97	0.97
Pollen	**0.95**	0.74	0.91	0.9	**0.95**	0.87
Usoskin	0.44	0.28	0.89	0.89	0.87	0.8

Table 2: NMI results for various preprocessing/clustering methods on different datasets. In most datasets, a method using UNCURL was either the best method, or tied as the best method.

4.2 Comparison of semi-supervision on 10x pooled data

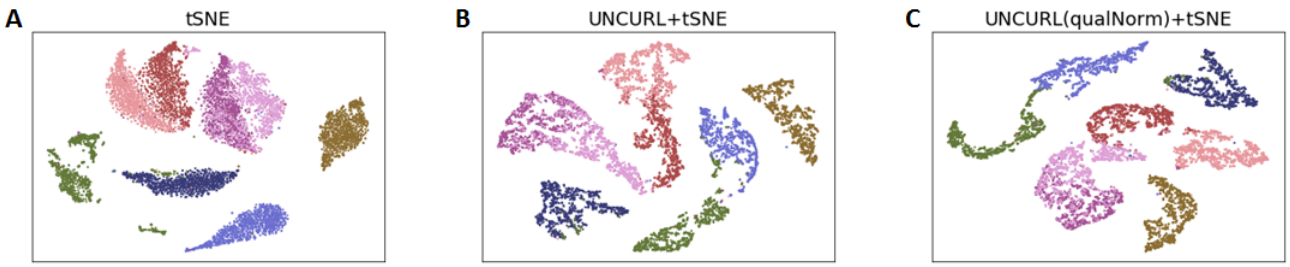

Figure 1: Effect on tSNE based visualization for 10x pooled dataset using different initialization strategies. A) tSNE without pre-processing. B) Unsupervised UNCURL + tSNE. C) QualNorm semi-supervised UNCURL + tSNE.

4.3 Comparison of lineage inference on synthetic lineages

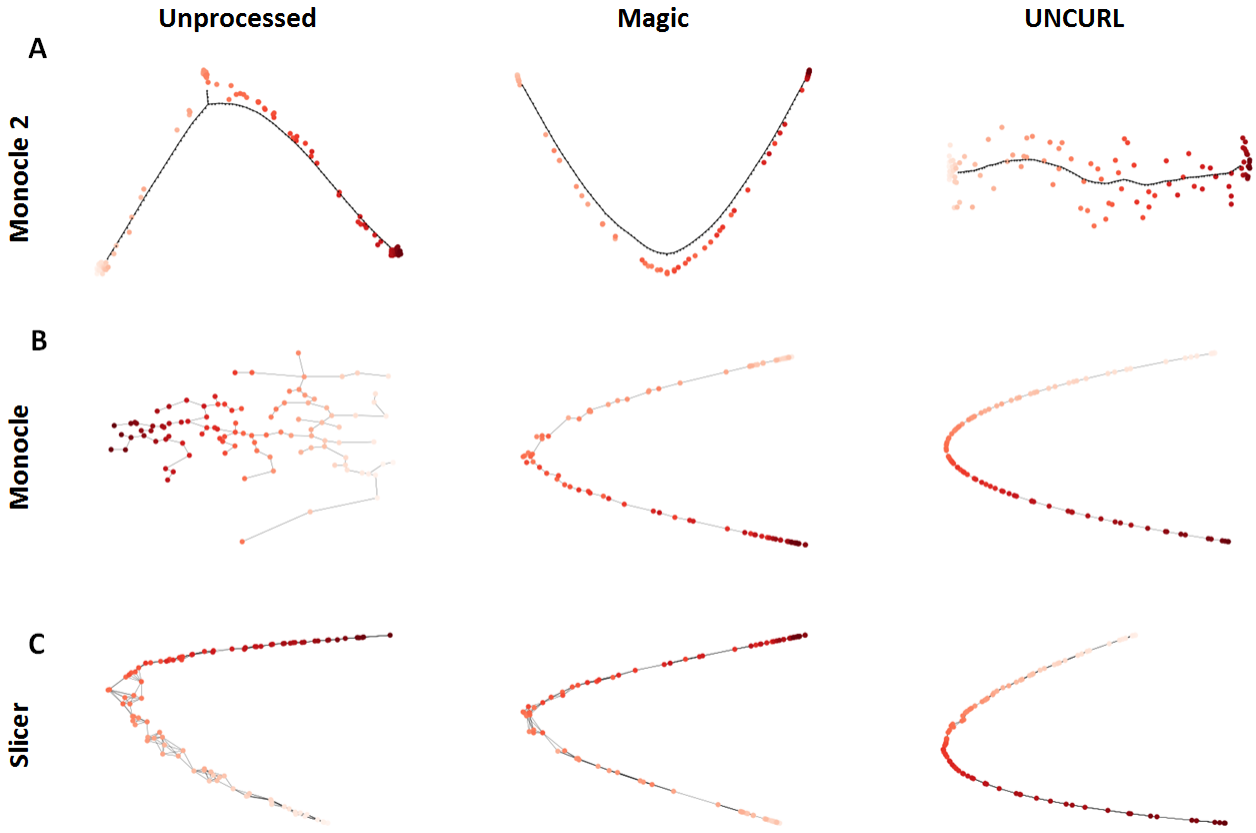

Figure 2: Comparison of lineage estimation on a synthetic linear lineage containing 100 cells. Comparison of different algorithms and different three different pre-processing methods namely A) Unprocessed, B) Magic pre-processed, C) UNCURL pre-processed. Cells are colored by true progress along the lineage.

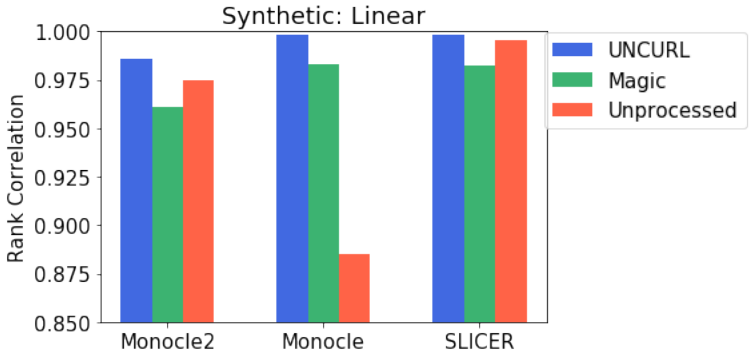

Figure 3: Rank correlation (with true pseudotime) of the pseudotime estimated by different lineage estimation algorithms using different pre-processing methods.

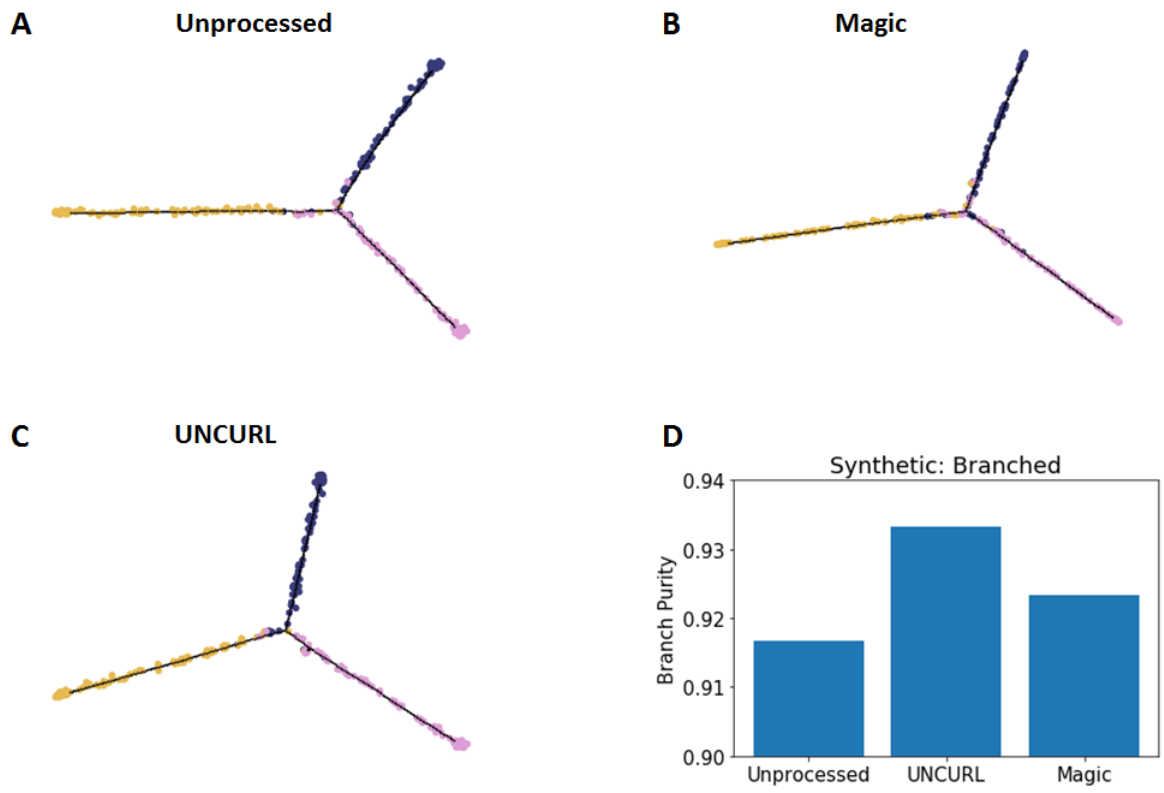

Figure 4: Estimated lineage along a branched trajectory (containing 300 cells, 100 per branch) using Monocle2 after different pre-processing methods. A) Unprocessed, B) UNCURL pre-processed, C) Magic pre-processed. D) Comparison of branch purity measured using identified branches and actual branches.

4.4 Unsupervised UNCURL lineage for Hanchate et. al.

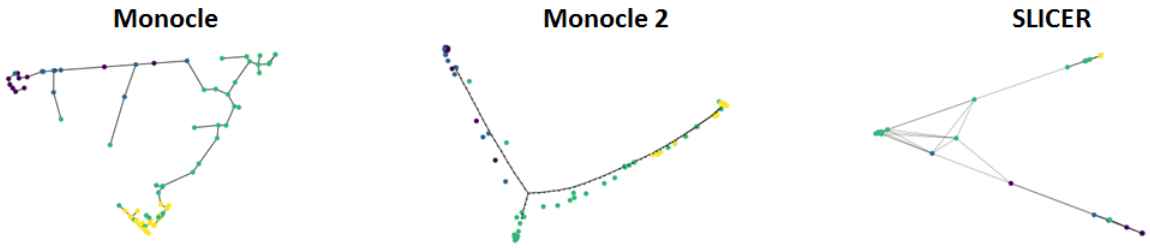

Figure 5: Unsupervised lineage estimation after UNCURL pre-processing using different methods on the dataset from Hanchate *et al.* (2015)

4.5 Timing comparison of various clustering methods

Dataset	Distribution	Cells	K	tSNE + kmeans	SIMLR	ZIFA + kmeans	Magic + tSNE + kmeans	UNCURL
10x_pooled	Poisson	73233	8	3615	2504	N/A	23188	**591**
10x_PBMC	Poisson	68579	10	4051	2733	N/A	9670	**490**
Zeisel	Poisson	3005	9	120	184	3040	93	**62**
Zeisel_sub	Poisson	3005	9	134	149	12778	96	**36.5**
10x_1.3M	Poisson	1.3 mil	10	87981.53	109887.2	N/A	N/A	**36869**
Tasic	Poisson	1629	49	**24**	27	2734	56	144
mESC	LogNorm	182	3	2	6.73	287	3	**0.5**
Kolod	LogNorm	704	3	9	84.75	2401	13	**2**
Pollen	LogNorm	249	11	2.7	10.5	964	6	**2.2**
Usoskin	Poisson	622	4	**8.3**	49.29	4603	21	19

Table 3: Comparison of run times of various algorithms on different single cell RNA-Seq datasets.

4.6 Parallelizability of UNCURL

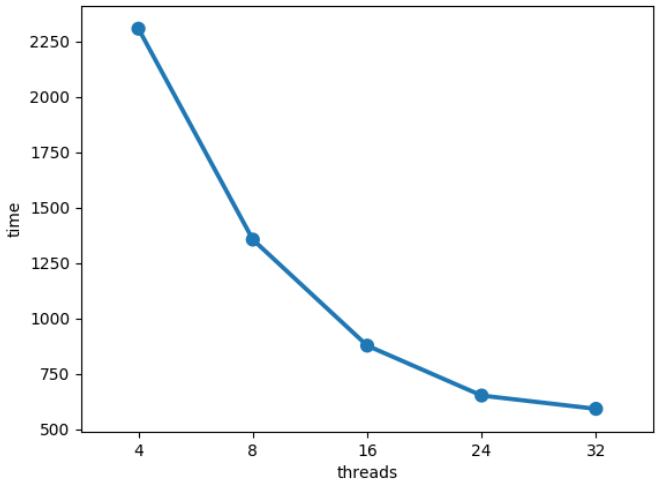

Figure 6: Comparison of run times for UNCURL with different number of computing threads for the 10x-pooled dataset. It is seen that the run times show an almost linear decrease till about 24 cores after which the performance saturates.

4.7 Robustness to choice of K

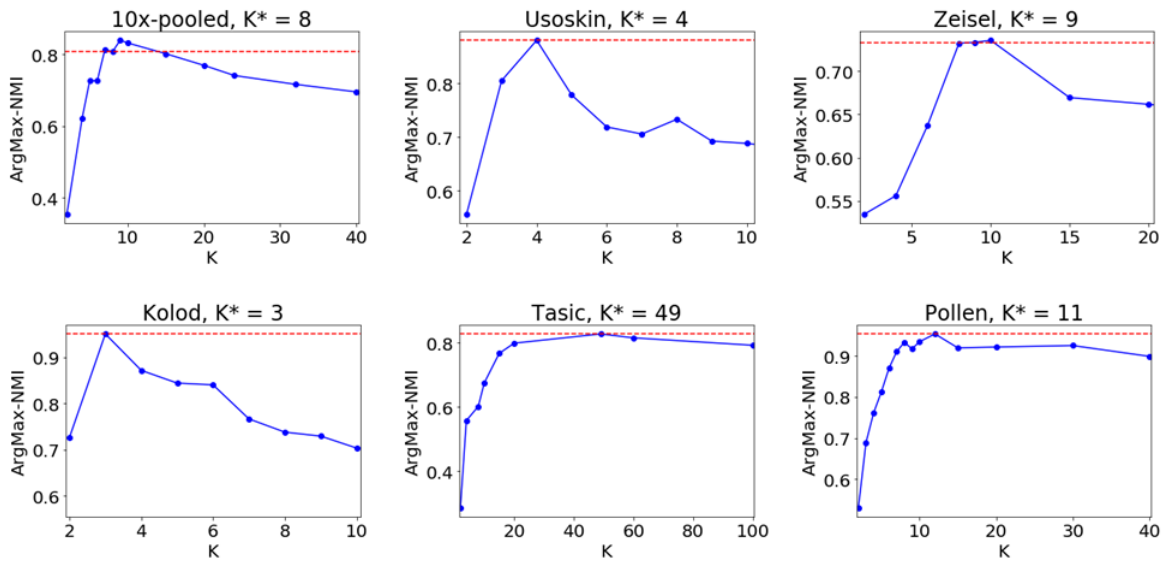

Figure 7: Robustness of NMI to choice of K. For a number of different datasets, we compared the NMI of arg-max(W) for different choices of K, where K* is the "correct" number of clusters. We see that while the NMI of UNCURL peaks around the true value of K, for most datasets, increasing it further leads to only a slight loss in precision. This shows that UNCURL is quite robust to the choice of K.

4.8 Cluster specific gene heatmap after Magic pre-processing for 20k subset 10x 1.3 million dataset

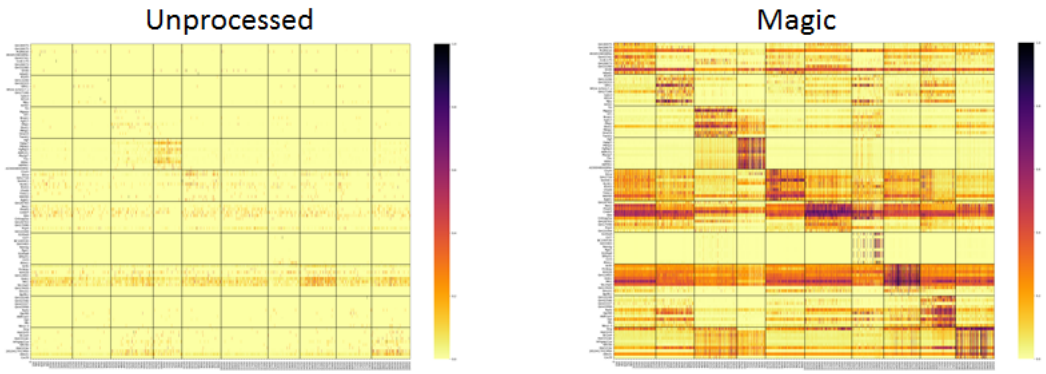

Figure 8: Clustered heatmaps showing the top cluster specific genes identified by Magic before and after pre-processing. Cells sorted by decreasing W for each cluster.

4.9 Cluster specific gene plots for 10x 1.3 million dataset

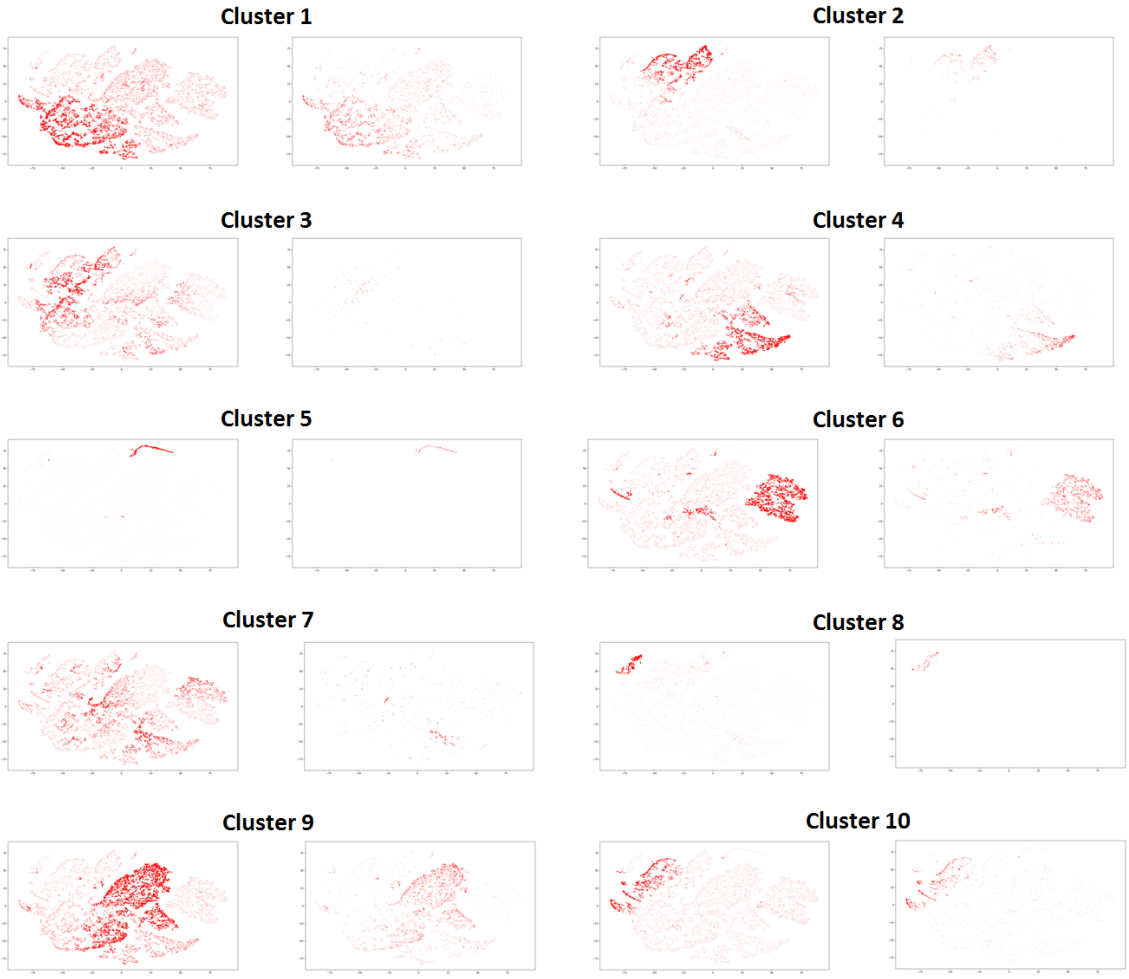

Figure 9: Average expression (with and without pre-processing) of the top cluster specific genes overlaid on the UNCURL processed tSNE plot.

4.10 Top cluster specific genes for 10x 1.3 million dataset

Cluster number	Genes
1	Dnah3, RP23-32A8.6, 1700013G24Rik, Gm11973, 1700101O22Rik, Gm20069, Gm11817, 1700042O10Rik, Tex21, Gm28856
2	Kcnk18, Paqr6, Rsph4a, 2210011K15Rik, Opn5, Gm42956, Tnn, Map3k19, Tctex1d1, Wnt3a
3	Krt18, Synpo2l, Gm43419, Gm13583, Prdm13, Hoxc6, Odf3l1, Ttll10, Pebp4, Gm26516
4	Myh2, Gm14637, Slc6a5, Dsg1a, Hormad1, Gys2, RP23-274K7.1, Rxfp3, Fam163a, Tbata
5	Gm5483, BC100530, Stfa2l1, S100a9, Ngp, Wfdc21, Retnlg, Gm5416, Epb42, Stfa2
6	Cntnap3, Isl1, Gm43998, Cacng3, Htr3a, Nxph2, Gm16315, Erbb4, Cntnap5c, Neb
7	Glycam1, Clca3a2, Ntn5, RP23-344G21.5, Trhr2, Gm8239, Cnn1, Slc9c1, Gm15294, Gm31728
8	Ccl7, Bcl3, Spint1, Il21r, Il1a, Ms4a7, Ptgs2, Tecrl, Srpx2, Gm5127
9	Crhr2, 1700080N15Rik, C8b, 4921521D15Rik, Baat, Arhgap33os, Macrod2os1, Smco1, Ly6g6f, Zdhhc23
10	Agbl1, Gm6116, Ttc39d, Dmrtc2, Gm20711, Btla, Cxcl9, 4933427J07Rik, Tbx5, Fbxw19

Table 4: Most highly overexpressed genes for each cluster identified in the 10x_1.M dataset

References

Buettner, F. et al (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature Biotechnology*, **33**(2), 155–160.

Hanchate, N.K. et al (2015). Single-cell transcriptomics reveals receptor transformations during olfactory neurogenesis. *Science*, page aad2456.

Heimberg, G. et al (2016). Low dimensionality in gene expression data enables the accurate extraction of transcriptional programs from shallow sequencing. *Cell systems*, **2**(4), 239–250.

Kolodziejczyk, A.A. et al (2015). Single Cell RNA-Sequencing of Pluripotent States Unlocks Modular Transcriptional Variation. *Cell Stem Cell*, **17**(4), 471–485.

Pollen, A.A. et al (2014). Low-coverage single-cell mRNA sequencing reveals cellular heterogeneity and activated signaling pathways in developing cerebral cortex. *Nature Biotechnology*, **32**(10), 1053–1058.

Tasic, B. et al (2016). Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature Neuroscience*, **19**(2), nn.4216.

Usoskin, D. et al (2015). Unbiased classification of sensory neuron types by large-scale single-cell RNA sequencing. *Nature Neuroscience*, **18**(1), 145–153.

Wang, B. et al (2017). Visualization and analysis of single-cell RNA-seq data by kernel-based similarity learning. *Nature Methods*, **14**(4), 414–416.

Zeisel, A. et al (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, **347**(6226), 1138–1142.

Zheng, G.X.Y. et al (2017). Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, **8**, 14049.