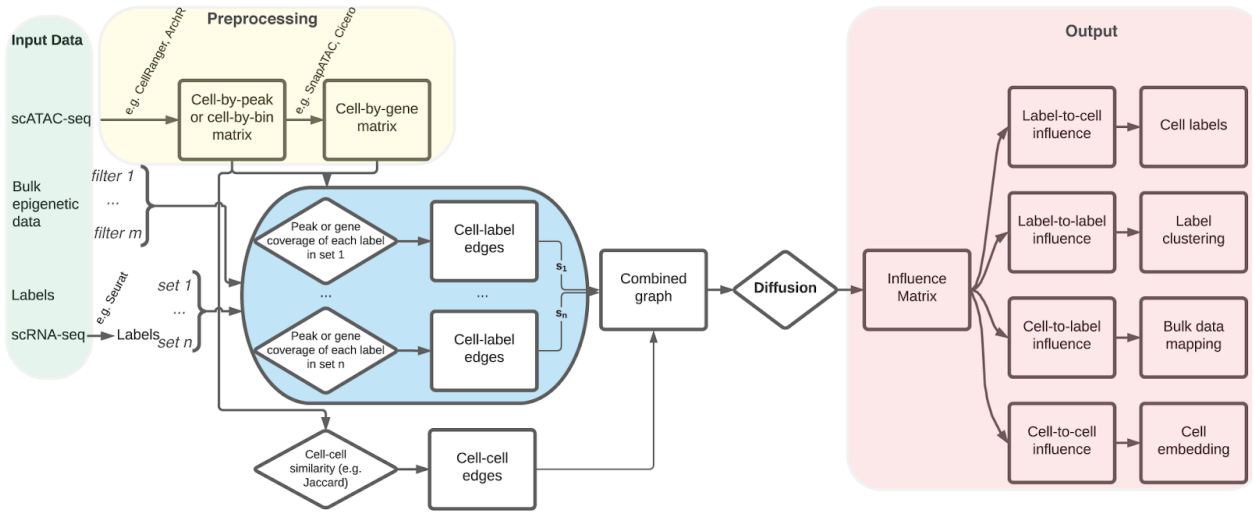
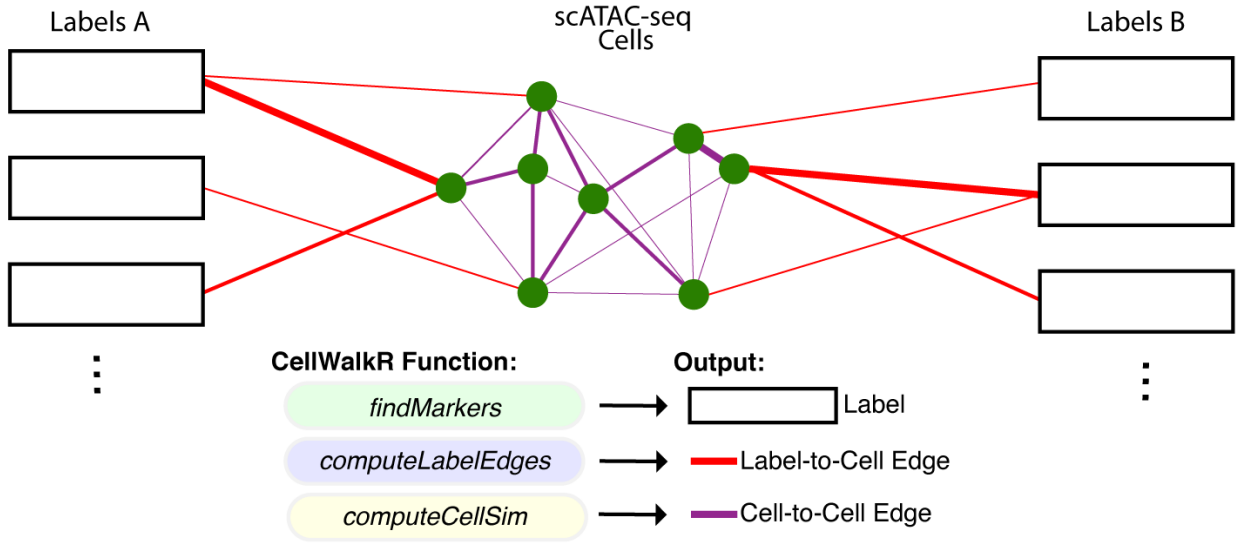
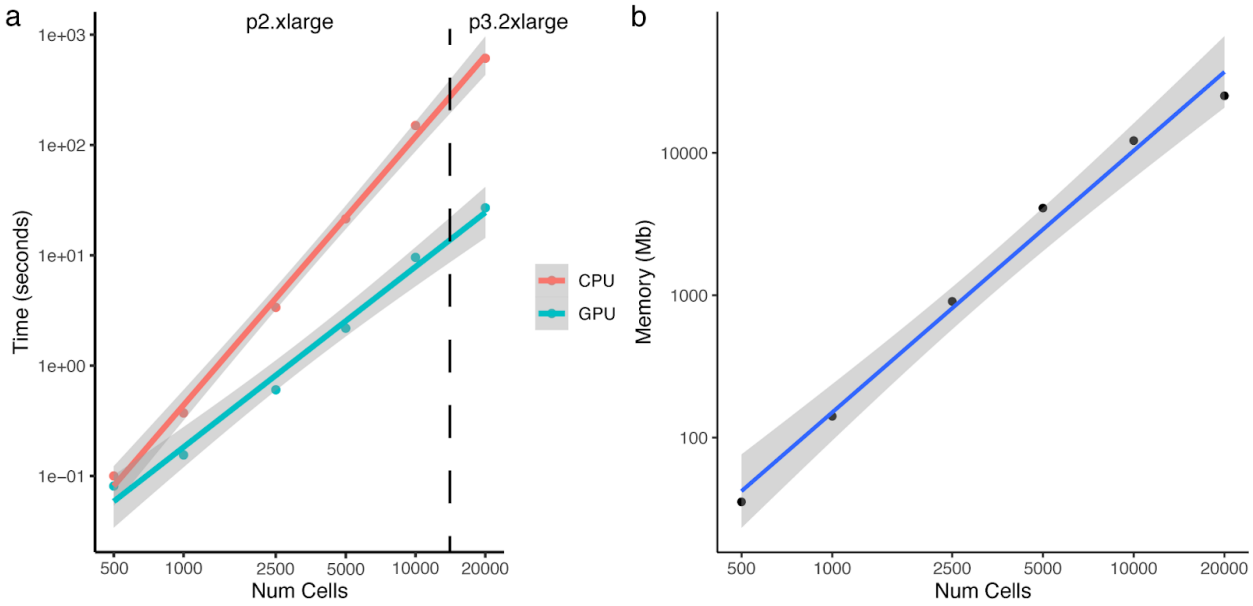


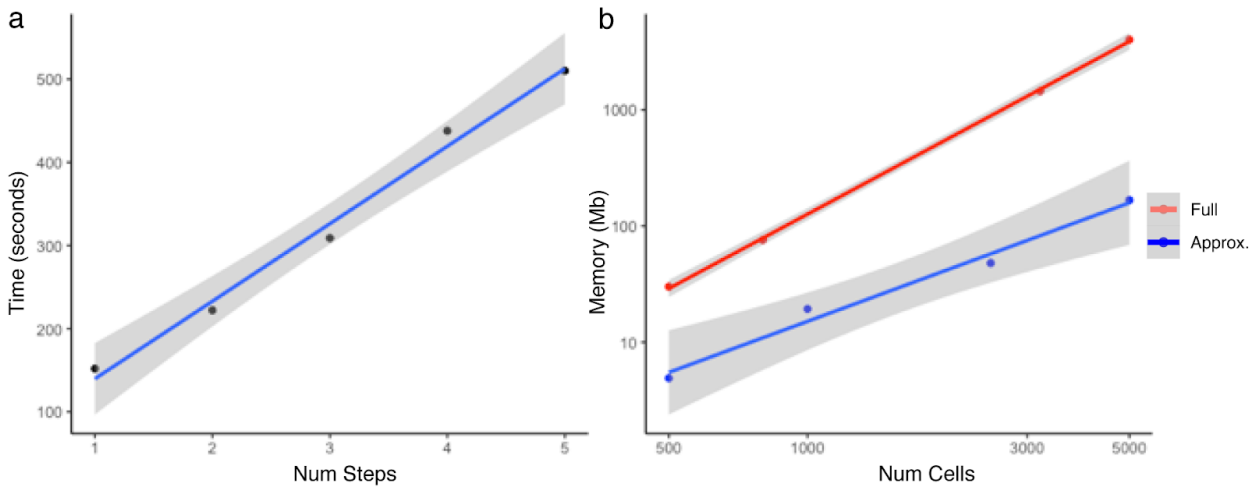
Supplemental Figure 1. **Flowchart of CellWalkR pipeline.** CellWalkR takes scATAC-seq, bulk epigenetic data, and label sets as input which it then combines into a graph. After diffusing information across this graph, different portions of the resulting influence matrix are used for analysis and visualization.



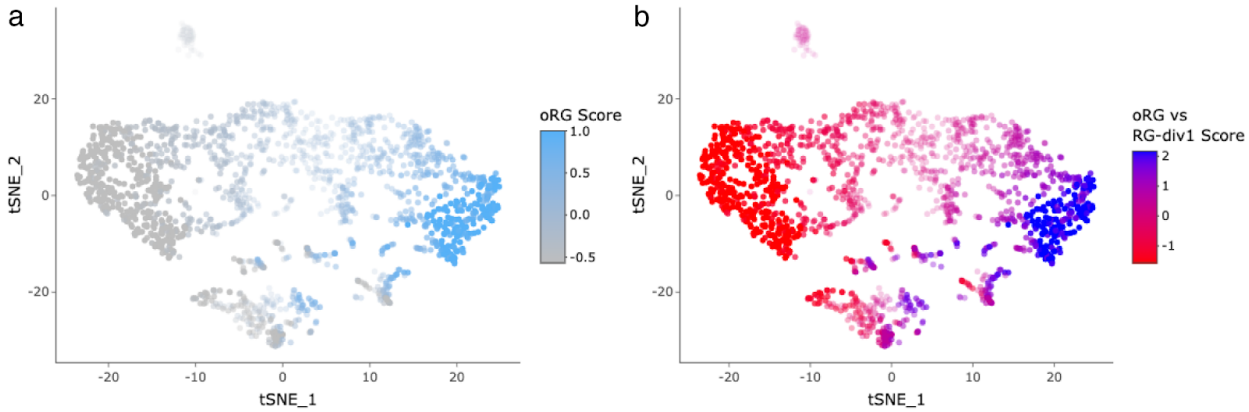
Supplemental Figure 2. **Network Constructed by CellWalkR.** CellWalkR builds a network consisting of two types of nodes representing cells and labels. The *findMarkers* function can be used to generate label nodes from scRNA-seq data. Edges between labels and cells are computed using the *computeLabelEdges* function, which can be supplied with multiple sets of labels. An edge is generated between a label and a cell based on how open the marker genes are in the cell's scATAC-seq data. The definition of regions used to quantify openness of marker genes is flexible and user-defined; it can correspond to promoters, gene bodies, and/or distal regions (e.g. correlated peaks identified by Cicero). Edges between cells are computed using the *computeCellSim* function, using Jaccard similarity by default, though any distance function can be passed.



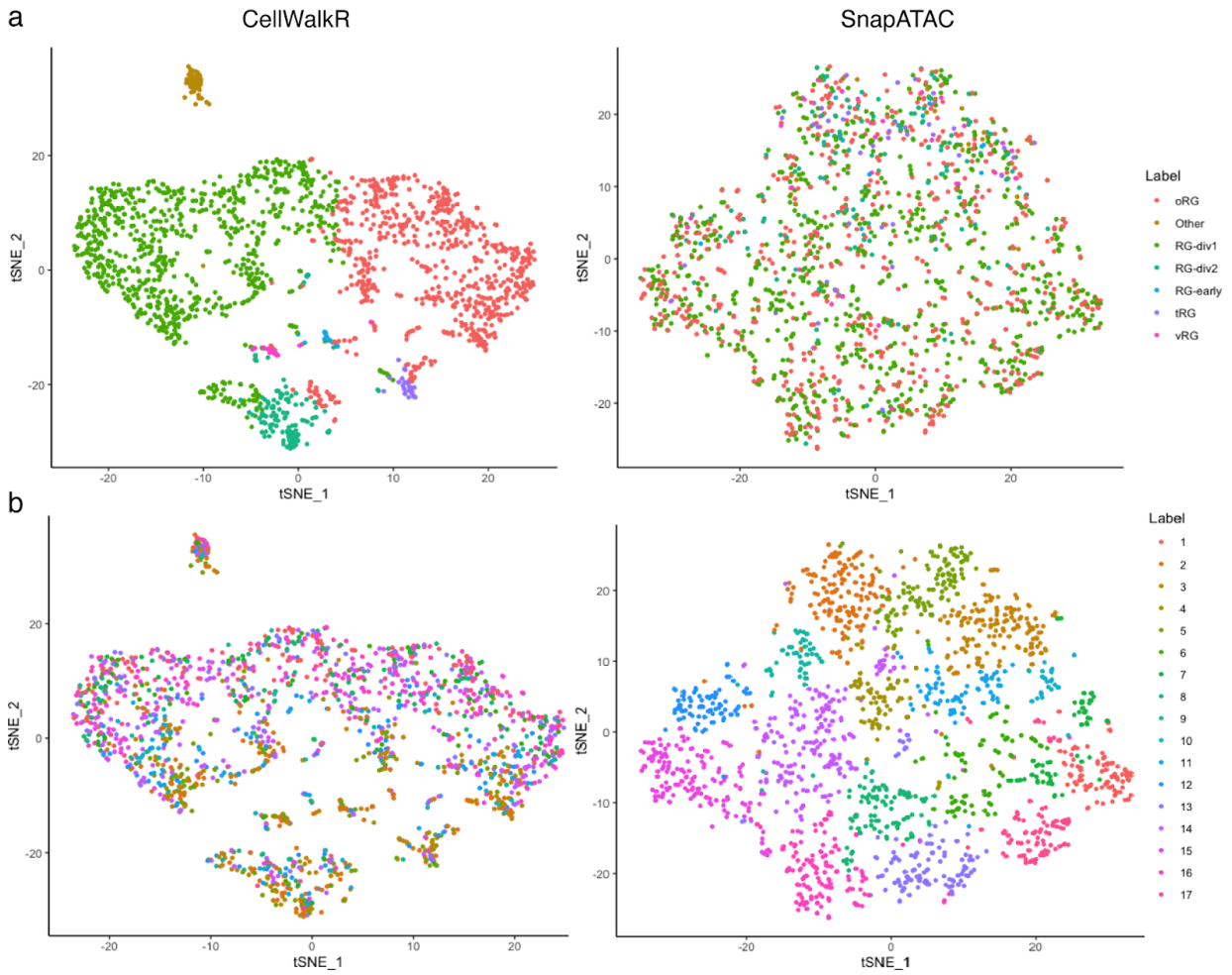
Supplemental Figure 3. **CellWalkR Performance. a)** Run time of CellWalkR on AWS P2.xlarge instance (for up to 15,000 cells) and P3.2xlarge (for 20,000 cells) using a CPU or GPU. A typical analysis of 10,000 cells can be run in 150 seconds on a CPU and 10 seconds on a GPU, while an analysis of 20,000 cells can be run in 611 seconds on a CPU and 27 seconds on a GPU. The GPU is more than 15 times faster, with greater benefits for more cells. **b)** Memory used by CellWalkR on a P2.xlarge instance. A typical analysis of 10,000 cells requires 12Gb of memory, while 20,000 cells requires 24Gb.



Supplemental Figure 4. **CellWalkR Performance With Approximate Solution.** The solution of a random walk with restarts can be approximated using an iterated walk, which requires consecutive matrix multiplications rather than a matrix inversion. Each step is computed as $F_{t+1} = \alpha I + (1 - \alpha)WF_t$ where *F* is the influence matrix and $W = D^{-1}A$ where *D* is a diagonal matrix of the degree of each node and *A* is the adjacency matrix. **a.** Run time of CellWalkR using a CPU to compute each step of the walk with 30,000 cells. For comparison, computing the full solution for 30,000 cells takes ~3 hours. **b.** Memory usage of CellWalkR to compute the full solution of the walk and five steps of the approximate solution. We extrapolate that computing the approximate solution for 30,000 cells requires 2Gb as compared to 170Gb for the full solution.



Supplemental Figure 5. **Plotting Label Scores a)** When a single cell type is selected, the amount of influence a single label has on each cell is shown on the generated t-SNE. **b)** When two cell types are selected, CellWalkR shows the difference in influence between two labels, allowing the user to identify transition regions.



Supplemental Figure 6. **Comparison to SnapATAC. a)** CellWalkR embedding of cell-to-cell influence using t-SNE (left) creates more distinct clusters than the embedding generated by SnapATAC (right) with clear separation of labeled cell types. Cells are colored by maximally influencing labels in CellWalkR, with cells receiving a maximum influence less than 0 marked as "Other." Note that labels are not defined by clustering in embedding, thus the grouping of cell types in the embedding can serve as a validation of the distinctness of labels. Additionally, CellWalkR displays an ability to identify both common cell types and very rare cell types, with a large dynamic range in cluster size. **b)** Using the same embeddings as in panel a, but now with cells colored by their cluster assignment according to SnapATAC. SnapATAC detects a very large number of clusters and has no built in ability to detect what cell type they represent. The clusters all include similar numbers of cells (right). When these cluster assignments are plotted in the CellWalkR embedding (left), there is a clear gradient from top to bottom but no separation between labels.