Reconstructing complex lineage trees from scRNA-seq data using MERLoT

R. Gonzalo Parra*[1], Nikolaos Papadopoulos*[1], Laura Ahumada-Arranz[1], Jakob El Kholtei[1], Barbara Treutlein[2], and Johannes Soeding[1]

*equal contribution
[1]Quantitative and Computational Biology Group, Max Planck Institute for Biophysical Chemistry, Goettingen, Germany
[2]Department of Evolutionary Genetics, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany

Advances in single-cell transcriptomics techniques are revolutionising studies of cellular differentiation and heterogeneity. However, reconstruction of cellular lineage trees with more than a few cell fates has proved challenging. We present MERLoT (`https://github.com/soedinglab/merlot`), a tool to reconstruct complex lineage trees from single-cell transcriptomics data. We demonstrate MERLoT's robustness, flexibility and capabilities on various real and simulated datasets.

Recent advances in single-cell sequencing techniques [1, 2, 3] permit to measure the expression profiles of tens of thousands of cells making ambitious projects like the single-cell transcriptional profiling of a whole organism [4] or the human cell atlas [5] possible. These efforts will better characterize the different cell types in multicellular organisms and their lineage relationships [6]. These advances also put within reach the question of how single cells develop into tissues, organs or entire organisms, one of the most fascinating and ambitious goals in biology that would also have wide-ranging consequences for the study of many human diseases.

A critical step is to develop methods that can reliably reconstruct cellular lineage trees that reflect the process by which mature cell types differentiate from progenitor cells. This is challenging due to the inherently high statistical noise levels in single cell transcriptomes, the high-dimensionality of gene expression space, and the strong non-linearities [6]. A cellular lineage tree does not only give us insight into the temporal order of branching and emergence of intermediary cell types. It can also substantially reduce the high noise levels by model-based averaging: Since many cells contribute to the idealized, smoothly varying expected gene expression rates that are imputed for every point on the tree, noise can be averaged out, and accuracy increases with the number of cells in the dataset.

A few methods exist for the reconstruction of lineage trees [7, 8, 9, 10]. All but the first start by embedding the set of high-dimensional gene expression vectors into a low-dimensional space, by which much of the noise is filtered out: SLICER uses local linear embedding [8], Monocle2 uses Reverse Graph Embedding (DDRtree) [9] and Destiny [10] uses diffusion maps [11].

We present MERLoT, a tool that can robustly reconstruct complex lineage trees. We first demonstrate MERLoT's performance on real datasets using different low-dimensional embedding methods. Our results on simulated data show that MERLoT's lineage tree reconstructions retrieve 18-95% more of the mutual information between predicted and known tree branch assignments, compared to the best next tool, Monocle2. Additionally, MERLoT can impute pseudotime courses [12] of gene expression levels along each trajectory on the lineage tree. This information can facilitate the analysis of gene regulatory networks controlling cell fate decisions and development.

Given a set of embedded coordinates for the cells, MERLoT reconstructs a lineage tree in three steps: (1) Calculation of a robust scaffold tree in the low-dimensional space combining the Dijkstra's shortest path [13] and Neighbor Joining [14] algorithms to define the location of endpoints, branchpoints and their connectivity; (2) smoothing and optimization of the scaffold tree using the Elastic Principal Tree (EPT) algorithm [15]; (3) embedding of the low-dimensional EPT into the high-dimensional gene expression space to impute pseudotime courses of gene expression profiles along the tree.

To build the scaffold tree, we select as first two endpoints the pair of cells that holds the maximum number of cells on their shortest path, and add those cells to the scaffold tree. We select as next endpoint the cell whose shortest path to any of the cells on the scaffold tree adds the most cells to it (Methods and Fig. S1). This process is iterated until new branches do not contain enough cells (auto mode, default threshold $= \sqrt{N}$, with N number of cells) or until a user-specified number of endpoints has been found (fixed mode). Subsequently, we apply the Neighbour Joining criterion to find the branchpoints and their connectivity with the endpoints (Methods and Fig. S2). When tested on many different datasets and embeddings, MERLoT's scaffold tree reconstructions proved to be very robust (Fig. 1A, B and C, Fig. S4).

Next, we use the scaffold tree to initialize an Elastic Principal Tree (EPT) [15] with it. The EPT algorithm fits a smooth tree to the cells, represented by connected, discretized support nodes. It minimizes the sum of squared distances of each cell to its nearest tree support node plus two terms enforcing smooth and unkinked trees. These terms allow us to interpolate through empty or low-density regions (Methods) and to aggregate in each support node information from many cells in its near and farther neigbourhood.

We applied MERLoT on three real datasets (details in Methods) (1) scRNA-seq data (with Unique Molecular Identifiers, UMIs) for myeloid progenitor differentiation (2730 cells, 3460 genes) [16], embedded in DDRTree coordinates (Fig. 1A,D), (2) single-cell qPCR data for zygote to blastocyst differentiation (428 cells, 48 genes) [17], embedded in a diffusion map (Fig. 1B,E), and (3) scRNA-seq data (index omics) for Haematopoietic Stem and Progenitor Cells (HSPCs) (1034 cells, 469 genes), using STEMNET coordinates [18]. The number of endpoints of the lineage tree, found by MERLoT in auto mode, are consistent with what has been reported in the original articles.

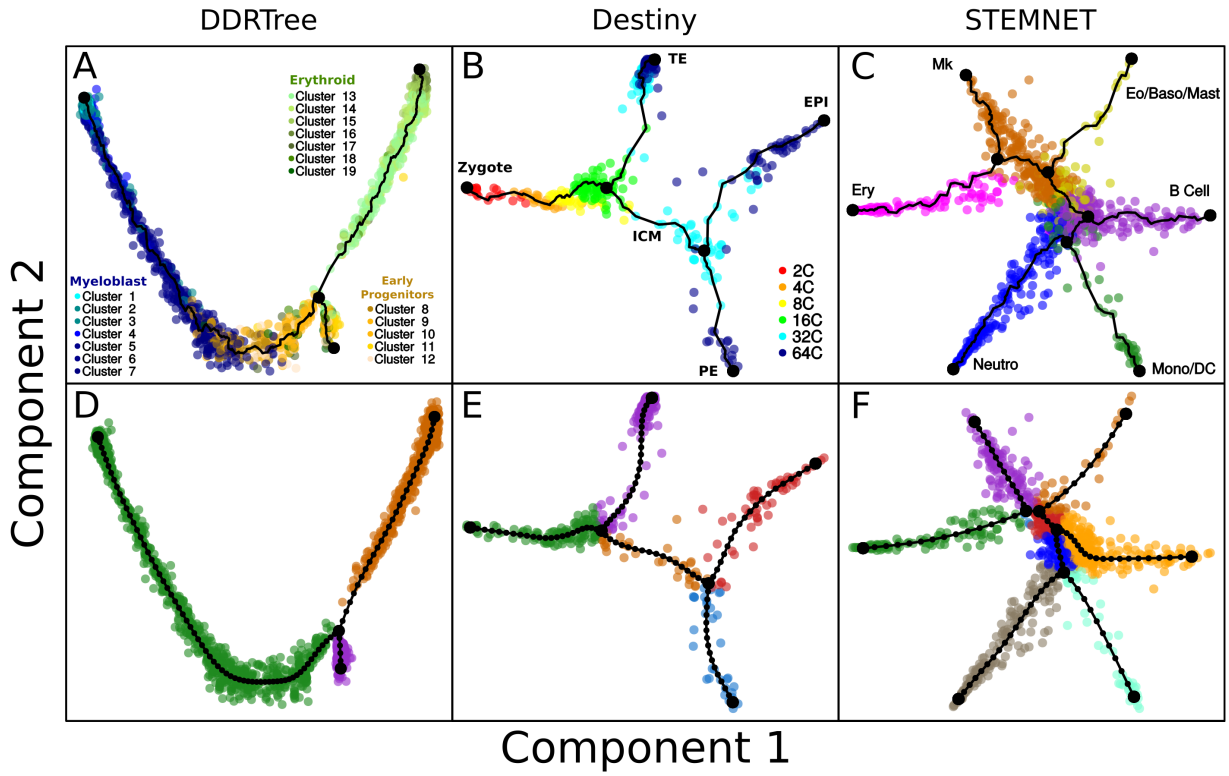

Figure 1: MERLoT's scaffold tree reconstructions: in combination with **(a)** DDRTree coordinates for analyzing the data from Paul et al. [16]; **(b)** diffusion maps coordinates for analyzing the data from Guo et al. [17] (this data corresponds to 3D data with coordinates 2 and 3 being rotated around coordinate 1 in order to produce a 2D representation of the data); **(c)** STEMNET coordinates for analyzing the data from Velten et al. [18]. Cells are colored according to cell type annotations provided by the authors of each dataset. **(d)**, **(e)** and **(f)** EPT reconstructions using the scaffold trees above as an initialization point. Cells are colored according to MERLoT's branch assignments.

After the lineage tree reconstruction, each node on it will be assigned a pseudotime value equal to the number of edges that separate it from the initial point, t_0. Subsequently, each cell will be assigned the pseudotime value of its closest tree node. In Fig. 2A, pseudotime values for the zygote to blastocyts dataset are shown. Because scRNA-seq data contain much technical and biological noise [6], cells with similar pseudotime values may have large variations in gene expression. MERLoT imputes denoised gene expression profiles by embedding the reconstructed lineage tree into the original gene expression space. Each tree support node in the low-dimensional space is mapped one-to-one to a tree support node in the gene expression space: We first assign each cell in the low-dimensional space to its nearest support node. Then, we initialize the corresponding support node in the gene expression space to the average gene expression level of all cells assigned to it (Fig. 2B top), and we run the EPT algorithm again (Methods). In this way, we impute the gene expression values of all the support nodes. These can be considered as pseudo cells representing idealized cell differentiation paths (Fig. 2B bottom, Fig. 2C). This model-based imputation results in denoised pseudotime courses of gene expression for the entire tree. As an example, Fig. 2C shows the expression profiles of four genes that are differentially expressed between EPI (red) and PE (blue) cell lineages (Methods). Note how the expression values of the pseudo cells (solid lines) interpolate and smooth the noisy single-cell expression values (circles).

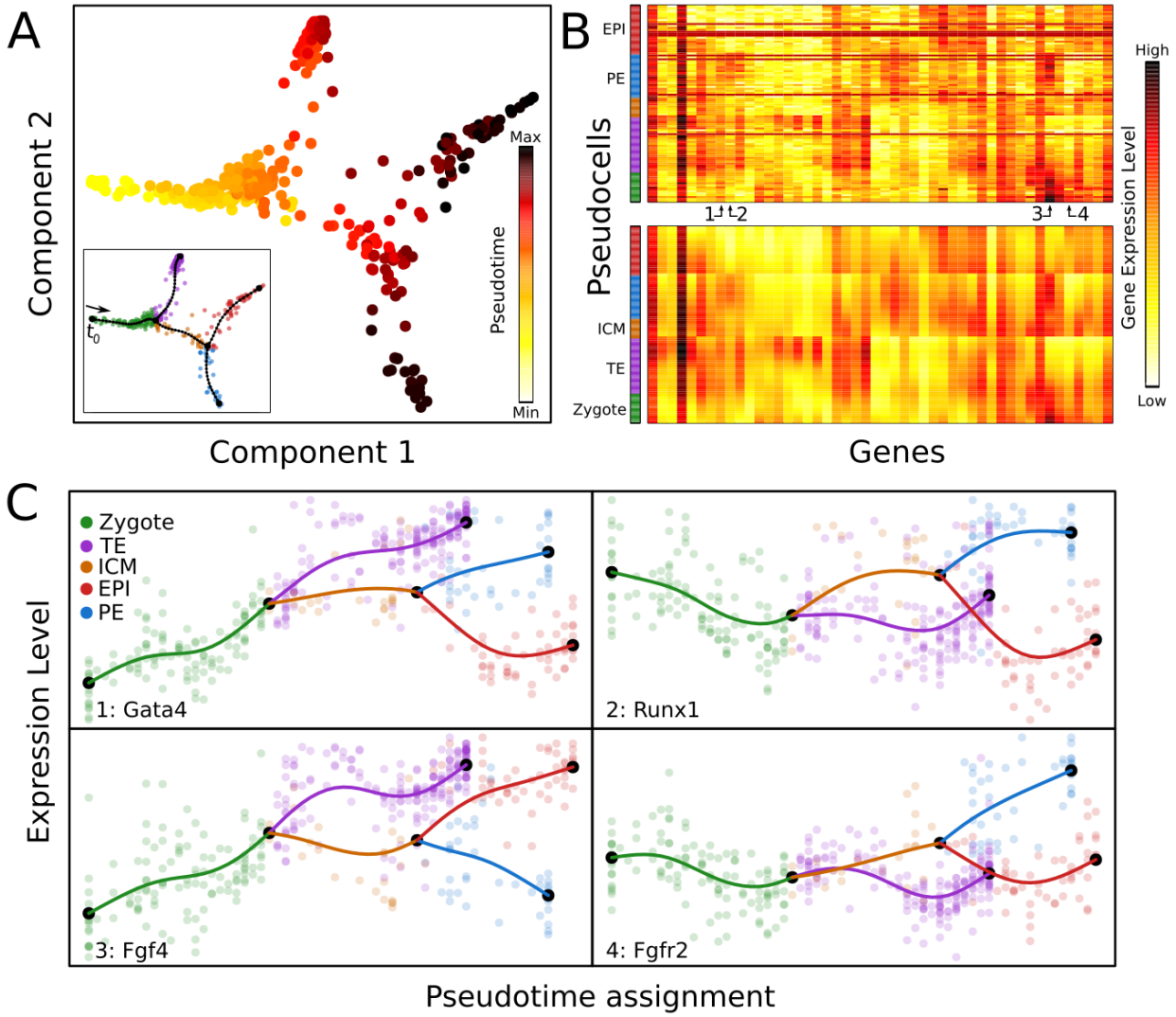

Figure 2: Pseudotime assignment and imputation of gene expression profiles: (a) Pseudotime assignment in color code for cells from Figs 1B and 1E, taking the zygote state as t_0. **(b)** Color-coded matrix of gene expression values for tree pseudo cells (rows) times genes (columns) before (top) and after (bottom) the gene expression space interpolation using the EPT. Pseudo cells are ordered according to their pseudotime and genes were hierarchically clustered. Numbers indicate specific genes shown in panel C. The color code in the bar on the left side of the heatmaps refers to pseudo cells branch assignments. **(c)** Gene expression profiles over pseudotime for 4 genes that are differentially expressed between the EPI and PE lineages. Semi-trasparent circles represent the expression values of individual cells and solid lines are the interpolations calculated by the EPT algorithm. Colors as in Fig. 1E.

We quantitatively estimated the performances of SLICER [8] and Monocle2 [9] to infer lineage trees from scRNA-seq data. Due to the lack of a gold standard set of real or simulated datasets on which to assess these tools, we developed PROSSTT (Methods, [19]), a software that can simulate scRNA-seq expression matrices with complex lineage tree structures. PROSSTT provides pseudotime and branch assignment labels for the cells, and branch connectivity information. Examples of diffusion maps for PROSSTT simulations and their lineage tree reconstructions performed by MERLoT are shown in Fig. 3A.

We generated a set of 1000 simulations divided into subsets of 100, each containing trees with 3 to 12 different cell fates and 1 to 10 bifurcations (Methods). We reconstructed the underlying lineage trees and assigned cells to branches using SLICER, Monocle2 and four flavours of MERLoT with diffusion maps or DDRTree coordinates, with and without providing the correct number of cell fates (fixed and auto modes).

For branch assignment (Fig. 3B), we assessed the agreement between predicted and labeled assignments in the simulations, using the Normalized Mutual Information (NMI) (Fig. 3B) and other scoring measures (Methods and Fig. S4). NMI rewards grouping cells in the same branch when they belong together and penalizes splitting them up. MERLoT using fixed mode, combined with diffusion maps, captures almost twice as much NMI as Monocle2, the second

best method. Supplementary Fig. S5 shows branch assignments in color on the simulated trees, giving an intuitive sense of the large differences in quality between the MERLoT's branch assignments and the ones of Monocle and SLICER. NMI increases for more complex topologies since it gets harder to missassign cells between branches that are far apart in the tree. MERLoT's performance in auto mode decreases, because its default threshold to add branches is too strict and hence several branches are missed as the topology complexity increases. Monocle2 has a stable performance for all topology complexities, comparable to MERLoT when using DDRTree coordinates. SLICER shows a low performance mainly because the recursive branch assignment function stalls for many datasets, especially for complex topologies.

In a multi-branched lineage tree, multiple trajectories exist between progenitors and differentiated cell fates. Pseudotime imprints only a partial ordering which can only be assessed within each trajectory and pseudotimes values are not comparable between non-consecutive tree branches. We test pseudotime orderings on the longest path between cells with the lowest and highest pseudotime value according to the topology assigned by PROSSTT. We use the Goodman-Kruskal (Fig. 3C) and other indexes (see Methods and Fig. S6) as a measure of concordance between the true and predicted orderings along the longest trajectory. Although predictions for simple topologies show marked differences for the different methods, as the topology complexity increases, they all converge to similar performance values, except for Destiny's Diffusion Pseudotime (DPT), which becomes better beyond eight bifurcations.

MERLoT outperforms other lineage tree reconstruction methods leading to an NMI increase of up to 95% mainly because of the high performance of the Scaffold Tree (Fig. S4), which performs almost as well as the full method. In terms of pseudotime assignments MERLoT is only outperformed by Destiny's DPT assignments for datasets with more than 6 bifurcations. However, MERLoT can be easily combined with DPT pseudotime orderings.

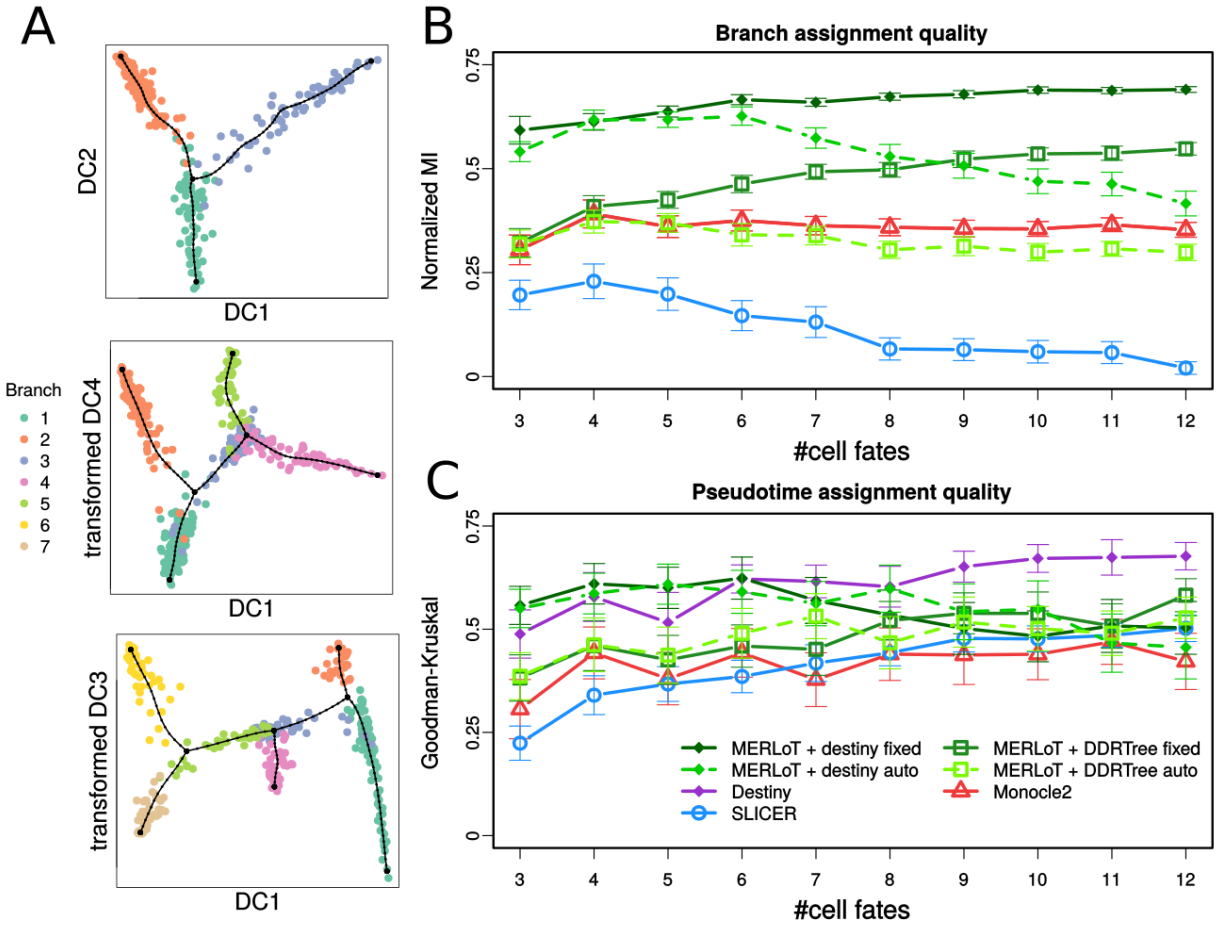

Figure 3: Simulated datasets and benchmarking: (a) Examples for diffusion map embeddings for simulations with one, two and three bifurcations. Cell colors represent PROSST labeled branch assigments. The double and triple bifurcation simulations correspond to 3D data. Double bifurcation: Diffusion coordinates 1, 2 and 3 were used. Coordinates 1 and 2 were rotated around coordinate 3. Triple bifurcation: Diffusion coordinates 1, 3 and 4 were used. Coordinates 3 and 4 were rotated around coordinate 1. **(b)** Branch assignment comparison using Monocle2, SLICER and 2 x 2 flavours of MERLoT combining Diffusion maps and DDRTree coordinates with the fixed and auto modes. **(c)** Pseudotime assignment comparison using Monocle2, SLICER, Destiny and the same four MERLoT versions as in panel B.

The single-cell RNA sequencing field is growing rapidly. Tools like MERLoT will be of great importance to study these challenging datasets and gain insights into the gene regulatory networks governing cellular differentiation and development

under healthy and disease conditions. By its model-based averaging, the limitations set by the high noise-level in single-cell transcriptomes can be overcome, opening up the way for quantitative modeling of the gene regulatory networks underlying complex developmental processes.

Methods

Methods, including statements of data availability and any associated accession codes and references, are available in the **online version of the paper**.

References

[1] Klein, A. M. *et al.* Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell* **161**, 1187–1201 (2015).

[2] Macosko, E. Z. *et al.* Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell* **161**, 1202–1214 (2015).

[3] Trapnell, C. Defining cell types and states with single-cell genomics. *Genome Res* **25**, 1491–1498 (2015).

[4] Cao, J. *et al.* Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science* **357**, 661–667 (2017).

[5] Regev, A. *et al.* The human cell atlas. *bioRxiv* https://doi.org/10.1101/121202 (2017).

[6] Wagner, A., Regev, A. & Yosef, N. Revealing the vectors of cellular identity with single-cell genomics. *Nat Biotechnol* **34**, 1145–1160 (2016).

[7] Herring, C. A. *et al.* Unsupervised trajectory analysis of single-cell rna-seq and imaging data reveals alternative tuft cell origins in the gut. *Cell systems* (2017).

[8] Welch, J. D., Hartemink, A. J. & Prins, J. F. Slicer: inferring branched, nonlinear cellular trajectories from single cell rna-seq data. *Genome Biol* **17**, 106 (2016).

[9] Qiu, X. *et al.* Reversed graph embedding resolves complex single-cell trajectories. *Nat Methods* **14**, 979–982 (2017).

[10] Angerer, P. *et al.* destiny: diffusion maps for large-scale single-cell data in r. *Bioinformatics* **32**, 1241–1243 (2016).

[11] Coifman, R. R. *et al.* Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc Natl Acad Sci U S A* **102**, 7426–7431 (2005).

[12] Trapnell, C. *et al.* The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells.. *Nat Biotechnol* **32**, 381 (2014).

[13] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik* **1**, 269–271 (1959).

[14] Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol and Evol* **4**, 406–425 (1987).

[15] Gorban, A. & Zinovyev, A. Elastic principal graphs and manifolds and their practical applications. *Computing* **75**, 359–379 (2005).

[16] Paul, F. *et al.* Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell* **163**, 1663–1677 (2015).

[17] Guo, G. *et al.* Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst. *Dev Cell* **18**, 675–685 (2010).

[18] Velten, L. *et al.* Human haematopoietic stem cell lineage commitment is a continuous process. *Nat Cell Biol* **19**, 271 (2017).

[19] Papadopoulos, N., Parra, R. G. & Soeding, J. Prosstt: probabilistic simulation of single-cell rna-seq data for complex differentiation processes. *bioRxiv* (2018) https://doi.org/10.1101/256941.

Online Methods

1 Lineage Tree Reconstruction by MERLoT

Given an expression matrix with N cells as rows and G genes as columns, a manifold embedding can be performed using techiques like Diffusion Maps or DDRTree embedding. Subsequently, informative coordinates can be kept in order to reduce dimensionality. Once cells are embedded into the low dimensional space, MERLoT can perform a lineage tree reconstruction following three steps which will be detailed in the following sections: (1) Calculating a Scaffold Tree with the location of endpoints, branchpoints and their connectivity. (2) Smoothing of the Scaffold Tree by using an Elastic Principal Tree (EPT) in the low dimensional manifold. (3) Embedding the EPT into the high-dimensional gene expression space and assigning pseudotime values to the cells.

Scaffold Tree Reconstruction

Matrix of shortest paths between cells: We used the Dijkstra's algorithm as implemented in the `csgraph` module from the `scipy` python library to calculate the shortest paths between all pairs of cells i and j based on their squared Euclidean distance d_{ij}^2. We use d_{ij}^2 instead of d_{ij} given that in a fully connected graph the shortest path between two nodes, based on d_{ij}, corresponds to the direct path between them. The `csgraph shortest_path` function returns the lenght D_{ij} of shortest path that connects cells i and j. We extended the function to also return the number of cells S_{ij} on the shortest path connecting cells i and j. The modified code for `csgraph` is available at `github.com/soedinglab/csgraph_mod`.

Endpoints search: MERLoT does not require users to define a starting point in order to reconstruct the tree topology. The first two endpoints in the tree correspond to the pair of cells k, l that maximizes the number of nodes S_{kl} on their shortest path (Fig. S1A): $(k, l) = \arg\max\{S_{kl} \,:\, 1 \leq k \leq l \leq N\}$. In case of ties, the pair of cells with the longest shortest path distance will be selected.

The next endpoint to be selected will correspond to the cell n that maximizes $s_{\mathcal{E}}(n)$, the number of cells being added to the scaffold tree structure. To compute $s_{\mathcal{E}}(n)$ we note that the new endpoint n must branch off from an internal node of the tree whose next-neighbour nodes must lie on the path of two already selected endpoints k and l. Therefore, the increase in number of cells is 0.5 times the minimum of the cells between k and l *via* n minus the cells between k and l, minimised over all pairs of endpoints (k, l) in the set of already selected endpoints $\mathcal{E}$.

$$s_{\mathcal{E}}(n) := 0.5 \times \min\{S_{kn} + S_{nl} - S_{kl} \,:\, k, l \in \mathcal{E}\}. \tag{1}$$

Stop criterion for endpoints search: In auto mode every time a new endpoint is selected we evaluate if $\max\{s_{\mathcal{E}}(n',) : 1 \leq n' \leq N, n' \notin \mathcal{E}\} > \sqrt{N/2}$ holds true. Otherwise, we calculate the branchpoints and tree connectivity for the endpoints in $\mathcal{E}$, including n, using the methodology explained in the next subsection. After this, all cells are mapped to their closest branch. If the branch added by the selected n endpoint contains more than `MinBranchCells`$= \sqrt{N}$ cells mapped to it, the branch is kept in the tree scaffold structure and the endpoints search is repeated. Otherwise, the endpoint search terminates and n is discarded as endpoint. The `MinBranchCells` threshold can be modified by the user. Alternatively, instead of using a stop criterium, users can set the number of endpoints that are aimed to be found (fixed mode) regardless of the branch lengths.

Branchpoints search and tree connectivity definition

Once endpoints are found, we apply the in combination with a heuristic criterion to find the cells that best represent the branchpoints in the tree structure. By doing this, we also detect the tree connectivity among endpoints and branchpoints

Given a set of endpoint and branchpoint nodes we iterate the following steps: (1) Use the Neighbour Joining (NJ) criterion [1] to select the pair of (k, l) nodes that will be joined *via* a branchpoint next. 2) Find the cell m that best represents the branchpoint between k and l and add the $l - m$ and $m - k$ edges to the tree (Fig. S2).

Let $\mathcal{V}$ be the set of yet unprocessed endpoint and branchpoint nodes of the tree. We initialise $\mathcal{V} \leftarrow \mathcal{E}$ with the endpoint set $\mathcal{E}$ determined in the previous subsection.

(1) Given the matrix D_{kl} of shortest path lengths between nodes in $\mathcal{V}$ (section 1.1), the NJ criterion allows us to pick two nodes k, l in $\mathcal{V}$ that are guaranteed to be next neighbours and therefore can be linked via a single branchpoint. The nodes to be joined are chosen such that $(k, l) = \arg\min D_{k,l}^{NJ}$ where d_{kl}^{NJ} is the neighbour-joining distance,

$$d_{kl}^{\mathrm{NJ}} := D_{kl} - \frac{1}{|\mathcal{V} - 2|} \sum_{m \in \mathcal{V}} (D_{mk} + D_{ml}) \,. \tag{2}$$

(2) We determine the branchpoint cell m as the one that minimizes the sum of distances to nodes k, l and the mean distances to all other nodes included in $\mathcal{V}$,

$$m = \arg_m \min \left\{ D_{km} + D_{lm} + \frac{1}{|\mathcal{V}-2|} \sum_{n \in \mathcal{V} \setminus \{k,l\}} D_{nm} \right\} . \tag{3}$$

Next, k and l are removed from $\mathcal{V}$ and the new branchpoint m is added to $\mathcal{V}$, $\mathcal{V} \leftarrow \mathcal{V} \setminus \{k,l\} \cup \{m\}$.. Also, the edges $l-m$ and $m-k$ are added to the tree (Fig. S2).

We iterate (1) and (2) until $|\mathcal{V}| = 2$, which means no further branchpoints exist. After termination, we have determined the tree topology with its $|\mathcal{E}|$ endpoints and $|\mathcal{E}|-2$ branchpoints, each represented by a cell. The same cell can be detected more than once as a branchpoint and hence bifurcations with higher orders than binary ones are possible.

Elastic Principal Tree in the Low Dimensional Manifold

To produce smoother, more homogenously interpolated lineage trees MERLoT uses the Elastic Principal Trees (EPT) algorithm [2] as implemented in the rpgraph module (`https://github.com/Albluca/rpgraph`). The EPT algorithm is used to approximate the distribution of cells in a given space with a tree structure composed of k nodes. Direct application of the EPT algorithm is unstable as it often returns trees that are manifastly far from the global optimum, e.g. wrong number of endpoints or grossly missplaced branchpoints). We initialize the EPT with the coordinates of a set of l nodes consisting of the endpoints and branchpoints in the scaffold tree and their connectivity. Further nodes are then added by the EPT algorithm by iterative bisection of edges until it reaches the specified number of k support nodes. This procedure cannot modify the number of endpoints and branchpoints specified at the initialization step.

In every iteration an energy potential defined as follows is minimized:

$$U = \text{MSE} + \mu U_E + \lambda U_R. \tag{4}$$

MSE is sum of squared distances of cells to their closest tree support nodes. U_E and U_R are two regularization terms that ensure that we learn trees with regularly spaced points and with edges without kinks, respectively.

We performed a grid search around the EPT default values in order to optimize μ and λ and visually examined the reconstructed EPTs on the datasets shown in Fig. 1. Using $k = 100$, we obtained $\mu_0 = 0.00625$ and $\lambda_0 = 2.03e-09$. If the number of nodes used for calculating the EPT is changed, μ and λ are adjusted according to $\mu = (k-1)\mu_0$ and $\lambda = (k-2)^3\lambda_0$. All reconstructions in our benchmark were performed with the standard function using $k = 100$. However, for some particular topologies μ and λ might need to be tuned in order to produce optimal results, in particular if k is increased a lot. Alternatively, MERLoT can bisect the edges in a given EPT, by additional nodes producing a new EPT with almost $2k$ support nodes. Note that these are special cases.

Elastic Principal Tree Embedding into the Gene Expression Space

First, cells are assigned to their closest support node. Next, every node is assigned to the average expression profile of all cells assigned to it. If a given node is not the closest node to any cell in the set, a null vector is assigned to it. In this way, we translate the positions of cells in the low-dimensional manifold space to approximate positions in the full gene expression space by constructing k pseudocells with averaged values for each gene coordinate.

Finally, the EPT algorithm is initialized with the average expression profiles and a list of edges representing their connectivity in the low dimensional EPT to calculate an EPT in the high-dimensional gene expression space.

Pseudotime Assignment

Pseudotime is a quantitative measure of the progress of a cell through a biological process [3]. Given the reconstruction of a lineage tree by MERLoT, cells can be assigned pseudotime values as a function of the number of edges along the structure that separate them from the initial point of the process. MERLoT automatically sets the initial pseudotime, t_0, to one of the first two detected endpoints. Users can also set t_0 to any endpoint, branchpoint or to any individual cell. In the latter case, the closest node to that cell will be assigned as t_0 and the pseudotime values for the other nodes will be assigned as before. After a pseudotime value is assigned to each support node, cells will take the pseudotime value from their closest node in the tree. MERLoT can calculate pseudotime in both the low-dimensional manifold space and in the high-dimensional gene expression space.

2 Benchmark on Synthetic Datasets

2.1 Simulating count data of branching processes with PROSSTT

To evaluate method performance on reconstructing lineage tree structures we used our tool PROSSTT [4] to simulate 10×100 scRNA-seq datasets with topologies with between 1 and 10 branchpoints (3-12 endpoints).

PROSSTT generates a simulated scRNA-seq dataset in four steps: (1) it generates a tree (number and length of branches, connectivity), (2) it simulates average gene expression levels $\mu_g(t, b)$ (pseudotime t, branch b), (3) it samples points in the tree (t, b) (4) it retrieves $\mu_g(t, b)$ for each sampled point and draws UMI counts from a negative binomial distribution

1. Generate tree: We sample the number of genes from a uniform distribution between 100 and 1000. These are typical numbers left after filtering out uninformative genes.

Each tree segment in every simulation had a pseudotime length of 50, corresponding to 50 cells on the branch (in homogeneous sampling model). This length allows the expression programs to diffuse enough to be distinct from each other. Starting at 2 bifurcations, alternative segment connectivity possibilities become available (Fig. S3)). We chose the lineage tree topology randomly for every simulation.

2. Simulate average gene expression along tree: PROSSTT models relative gene expression as a linear mixture of a small number of expression programs. For each tree segment, we simulate the time evolution of expression programs as a random walk with momentum term. Each simulation uses $K = 10$ expression programs. We use the weighted sum of only two randomly chosen programs per gene, with weights drawn from a beta distribution (parameters $a = 2, b = 2$). Program indices are picked with replacement, so some genes are only influenced by one program. A scaling factor (library size) was sampled for each cell and multiplied to the average gene expression values. We used a log-normal distribution with $\mu = 0$ and $\sigma = 0.8$.

3. Sample cells from tree: PROSSTT can sample cells in the tree according to a given density function. Here we simply drew one cell per pseudotime unit for all branches.

4. Simulate UMI counts: We simulate unique molecular identifier (UMI) counts using a negative binomial distribution. Following [5] and [6], we make the variance σ_g^2 depend on the expected expression μ_g as $\sigma_g^2 = \alpha_g \mu_g^2 + \beta_g \mu_g$. The α_g and β_g values were sampled from log-normal distributions with $\mu_{a_g} = 0.2, \sigma_{a_g}^2 = 1.5$ and $\mu_{b_g} = 2, \sigma_{b_g}^2 = 1.5$ respectively. These values are typical for single-cell RNA sequencing UMI counts. For more information about default parameters choices and the algorithm, please refer to [4].

2.2 Assessing Methods Performance

Tree reconstruction tools need to succeed at two intertwined tasks: to arrange all cells according to their internal developmental time, while also detecting and separating different branches of the tree. Ideally, one would evaluate algorithm performance on both tasks at once. However, we were not able to find such a measure and so evaluate branch assignment and pseudotime prediction separately.

Branch assignment We treated branch assignment evaluation as a clustering problem. We can consider all cells mapped to a given branch of a tree as a cluster and compare the set of labels produced by PROSSTT with those that are predicted by each algorithm.

We used four measures to assess method performance: the F1 measure, the Matthews Correlation Coefficient, the Jaccard Index and Adjusted Mutual Information (called Normalized Mutual Information, or NMI, here) (Fig. S4), which all produce similar results. In our opinion the NMI is best for assessment of branch assignment, since it captures the amount of information present in the original clustering that was recovered by the prediction (values between 0 and 1). It corrects the effect of agreement solely due to chance between clusterings by using a hypergeometric background distribution and punishes overbranching and merging branches almost equally [7]. Given the predicted and real cluster assignments U and V with R and C clusters respectively, NMI is defined as

$$\mathrm{NMI}(U, V) = \frac{\mathrm{MI}(U, V) - \mathbb{E}[\mathrm{MI}(U, V)]}{\max\{H(U), H(V)\} - \mathbb{E}[\mathrm{MI}(U, V)]}.$$

Pseudotime prediction Evaluating the performance of the different methods consists of quantifying the degree of agreement between the true/labeled and the predicted orderings provided by the different algorithms. Pseudotime only establishes a partial ordering on cells and no absolute time. Therefore, cells on branches not passed through one after the other cannot be compared. We find the longest path in the tree (from the root to a leaf) and compare the predicted pseudotime with the simulated one for the cells on this path.

We used the Goodman-Kruskall index and the Kendall index (weighted and unweighted) to assess pseudotime prediction, as proposed by [8]. All four indices count how many pairs of cells have been ordered correctly (S_+) or incorrectly (S_-) and produce similar results for the benchmark. The unweighted Goodman-Kruskall index is the simplest approach: $(S_+ - S_-)/(S_+ + S_-)$.

2.3 Benchmarked Methods and Parameters

Monocle2: Monocle2 (version 2.6.1) uses a reverse graph embedding (RGE) technique called DDRTree [9] to create a lineage tree and map points from the low-dimensional embedding of the tree to the original gene space. Based on the vignette for (`monocle_2.4.0`), we used the `negbinomial()` expression family for the data and the proposed defaults for lower detection limit (1), minimum expression (`detectGenes` function, 0.1), mean expression threshold (gene ordering, 0.5), and empirical dispersion threshold ($2 \cdot dispersion_fit$).

Generally, a successful dimensionality reduction will capture a topology with B bifurcations in its first $d = B + 1$ dimensions. While Monocle2 runs DDRTree by default on two dimensions, in our benchmarks we used $B + 1$ dimensions, as it performed better. DDRTree did not always return d dimensions; if fewer dimensions were provided we used the maximum possible number.

Monocle2 assigns a branch identity to each cell in the `State` column of the phenotypic data table (`pData`). It calculates pseudotime as distance from one of the endpoints of the lineage tree it produces. In order to pick the correct endpoint, we checked if the cell with simulated pseudotime $t_0 = 0$ was in an outer branch; if true, the corresponding endpoint was assigned t_0. In the rare event that it was placed in an inner branch, we used Monocle's chosen starting `State` for pseudotime calculation.

SLICER: SLICER uses locally linear embedding to perform dimensionality reduction and uses a neighbour graph to order cells according to their distance from a user-specified starting cell (pseudotime prediction). It uses geodesic entropy to recursively detect branches.

We used SLICER version 0.2.0 (commit cb1be8a) by following the instructions that accompany the software on its github page (https://github.com/jw156605/SLICER/). We used the software's gene selection process as-is and used the selected genes to determine the best k value for the LLE, with a $k_{\min}$ value of 5. Much like with Monocle2, using $d = B+1$ LLE dimensions (over the default 2) for a dataset with B bifurcations improved performance. We used the same k value for the creation of the low-dimensional k-nearest neighbour graph as we did for LLE. For every simulation, we used the cell with labeled pseudotime $t_0 = 0$ as the starting point of the `cell_order` function, which predicts pseudotime for each cell. Finally, we used the same start point for the branch assignment step (`assign_branches`). This step very often failed to execute; these cases were assigned the worst possible score of each branch assignment measure. The issue was reported to the authors (https://github.com/jw156605/SLICER/issues/7) but until the time of this writing was not resolved.

Destiny: Destiny produces a dimensionality reduction and a pseudotime prediction based on it. We used the destiny diffusion map space as input for MERLoT, and benchmarked destiny's Diffusion Pseudotime (DPT). We used destiny (version 2.6.1) as described in the Bioconductor vignette. First we normalized the input data by correcting for library size and then log-transformed them. The only free parameter is the number k of nearest neighbours. By default, destiny uses a heuristic to determine it. For a dataset with B bifurcations we gave destiny $d = B + 1$ dimensions of the diffusion map to determine pseudotime, following the same reasoning as with Monocle2 and SLICER. We retrieved pseudotime predictions from destiny by calling the `DPT` function on the diffusion map object and then using the dimension which corresponded to diffusion pseudotime distances from the cell with pseudotime $t_0 = 0$.

MERLoT: We ran different flavors of MERLoT on embeddings from different dimensionality reduction tools:

1. **MERLoT + diffusion maps:** destiny was run to produce diffusion maps for MERLoT. The same protocol as in section 2.3 was used, except for the selection of free parameter k, which was done by a simple optimization. For a simulated lineage tree with B bifurcations we tested values of k between 5 and 100 and kept the k value that maximized the drop-off after the $d + 1$'th eigenvalue of the diffusion map, where $d = B + 1$.

2. **MERLoT + DDRTree:** Monocle2 was run in order to produce DDRTRee cordinates. The number of coordinates to be used to reconstruct the lineage tree was selected as described in section 2.3.

Each of these two options was combined with two different ways in which MERLoT finds the number of endpoints in the dataset.

1. **MERLoT auto:** MERLoT is run without the specification of the number of endpoints in the tree. MERLoT will use its internal branch length heuristic to determine new branches. The algorithm stops finding new endpoints when the new branch aimed to be included in the tree structure does not contain more than $\sqrt{(N)}$ number of cells, with N being the total number of cells in the dataset.

2. **MERLoT fixed:** MERLoT is run with a specified number of endpoints (tree leaves) to be found. MERLoT will ignore its internal branch length heuristic and will keep searching for branches until it reaches the specified number of endpoints regardless of how many cells are mapped to them.

All 2×2 combinations were tested (MERLoT on diffusion maps auto/fixed and MERLoT on DDRTree coordinates auto/fixed).

3 Datasets

Myeloid progenitors differentiation (Paul, et al. Cell, 2015 [10])

This dataset was produced applying massively parallel single-cell RNA-seq (MARS-seq) which uses unique molecular identifiers (UMIs). After filtering those cells that passed the quality checks and selecting a set of informative genes we obtained a matrix of 2730 cells and 3459 genes. Originally the authors reported 3461 informative genes with some of them being incorrectly formated as dates, e.g 5-Mar, 4-Sep. We were able to correct the IDs of most of them to valid GeneIDs except two (IDs: 7-Sep and 2-Mar) which were excluded from the analysis. The script for calculating the DDRTree coordinates using Monocle2 and the subsequent lineage tree calculation (Fig. 1A and Fig. 1D) is available at: https://github.com/soedinglab/merlot/tree/master/inst/example/ExamplePaul2015.R.

Mouse zygote to blastocyst (Guo et. al, Developmental Cell, 2010 [11])

The dataset was produced by the Biomark RT-qPCR system and contains Ct values for 48 genes measured in 442 mouse embryonic stem cells at 7 different developmental time points, from the zygote to blastocyst [11]. The data was cleaned and normalized by following the vignette from the Destiny package. A total number of 428 cells and 48 genes were kept in the final expression matrix. A diffusion map was calculated using Destiny and the first three diffusion coordinates were used to calculate the lineage tree. We rotated the cells and tree nodes coordinates around the first axis in order to produce a two-dimensional representation of the data and improve visualization. The script for calculating the diffusion maps and the subsequent lineage tree calculation (Fig. 1B and Fig. 1E) is available at: https://github.com/soedinglab/merlot/tree/master/inst/example/ExampleGuo2010.R.

Haematopoietic Stem and Progenitor Cells (HSPCs) (Velten et. al, Nature Cell Biology, 2017 [12])

The scRNA-seq data was generated samples taken from two donor individuals. The authors used both smart-seq2.HSC (individual 1) and QUARTZ-seq (individual 2), and all findings were systematically compared between individuals. For obtaining the processed data we followed the vignette for the STEMNET software available as part of its R package. From that analysis we obtained the normalized data, the cell types labels and the STEMNET coordinates. The script for calculating the the STEMNET coordinates and the subsequent lineage tree calculation (Fig. 1C and Fig. 1F) is available at: https://github.com/soedinglab/merlot/tree/master/inst/example/ExampleVelten2017.R.

Differentially expressed genes detection

After a linear tree reconstruction has been performed, MERLoT can easily find groups of genes being differentially expressed among different groups of cells. If two groups of cells are provided, e.g cells assigned to two branches in the tree (2C)), MERLoT performs a Kruskall-Wallis rank sum test to evaluate which genes in the full expression matrix are differentially expressed on them. If a single subpopulation of cells is provided, the comparisson is made against the rest of cells in the data. The entire list of genes is given as output, ordered by the test p-values results. Also, e-values are provided by multiplying the p-values bby the number of G genes being tested.

Data availability

The 10 simulation sets with 100 simulated differentiations each are available at `http://wwwuser.gwdg.de/ compbiol/merlot/`. The code necessary to run the benchmark on the simulations as well as instructions about how to set up a similar benchmark are available at `https://github.com/soedinglab/merlot-scripts`. Formatted expression data for the three datasets in Fig. 1 are available at:
https://github.com/soedinglab/merlot/tree/master/inst/example/

References

[1] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425, 1987.

[2] Alexander Gorban and Andrey Zinovyev. Elastic principal graphs and manifolds and their practical applications. *Computing*, 75(4):359–379, 2005.

[3] Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. Pseudo-temporal ordering of individual cells reveals dynamics and regulators of cell fate decisions. *Nat Biotechnol*, 32(4):381, 2014.

[4] Papadopoulos, N., Parra, R. G. & Soeding, J. Prosstt: probabilistic simulation of single-cell rna-seq data for complex differentiation processes. *bioRxiv* (2018) https://doi.org/10.1101/256941.

[5] Dominic Grün, Lennart Kester, and Alexander Van Oudenaarden. Validation of noise models for single-cell transcriptomics. *Nat Methods*, 11(6):637, 2014.

[6] Kenneth D Harris, Carolina Bengtsson Gonzales, Hannah Hochgerner, Nathan G Skene, Lorenza Magno, Linda Katona, Peter Somogyi, Nicoletta Kessaris, Sten Linnarsson, and Jens Hjerling-Leffler. Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics. *bioRxiv*, 2017.

[7] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.

[8] Ricardo José Gabrielli Barreto Campello and Eduardo R Hruschka. On comparing two sequences of numbers and its applications to clustering analysis. *Information Sciences*, 179(8):1025–1039, 2009.

[9] Qi Mao, Li Wang, Steve Goodison, and Yijun Sun. Dimensionality reduction via graph structure learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 765–774, New York, NY, USA, 2015. ACM.

[10] Franziska Paul, Yaara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, David Lara-Astiaso, Meital Gury, Assaf Weiner, et al. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 163(7):1663–1677, 2015.

[11] Guoji Guo, Mikael Huss, Guo Qing Tong, Chaoyang Wang, Li Li Sun, Neil D Clarke, and Paul Robson. Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst. *Dev Cell*, 18(4):675–685, 2010.

[12] Lars Velten, Simon F Haas, Simon Raffel, Sandra Blaszkiewicz, Saiful Islam, Bianca P Hennig, Christoph Hirche, Christoph Lutz, Eike C Buss, Daniel Nowak, et al. Human haematopoietic stem cell lineage commitment is a continuous process. *Nat Cell Biol*, 19(4):271, 2017.

Supplementary Information
MERLoT: A method for reconstructing complex lineage tree topologies using scRNA-seq data

R. Gonzalo Parra*[1], Nikolaos Papadopoulos*[1], Laura Ahumada-Arranz[1], Jakob El Kholtei[1], Barbara Treutlein[2], and Johannes Soeding[1]

*equal contribution
[1]Quantitative and Computational Biology Group, Max Planck Institute for Biophysical Chemistry, Goettingen, Germany
[2]Department of Evolutionary Genetics, Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany

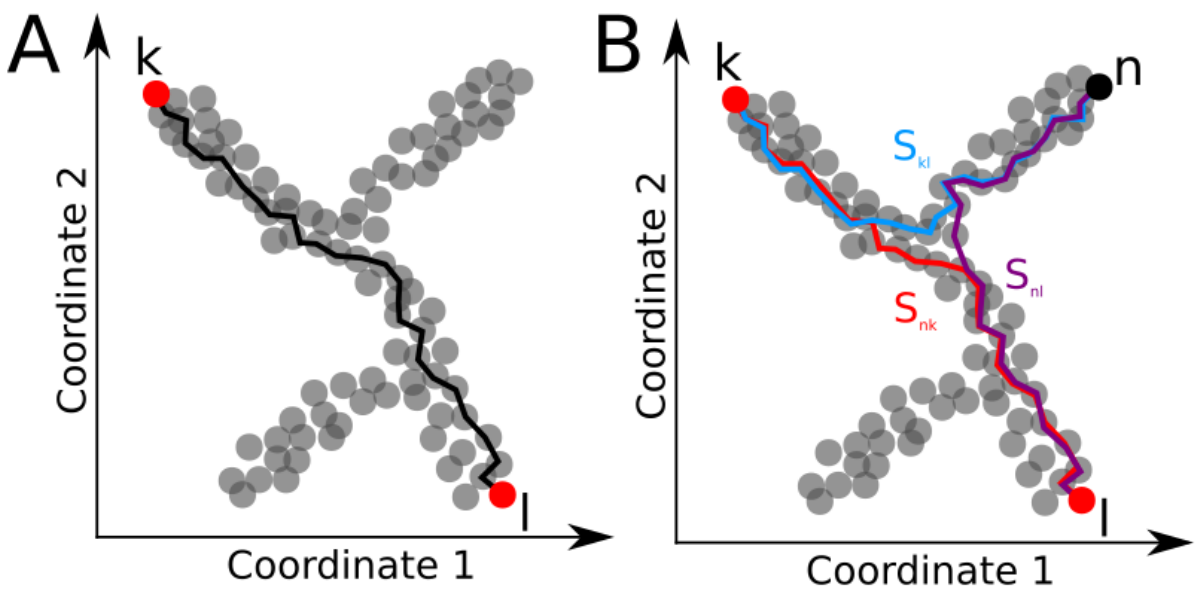

Figure S1: MERLoT's method for finding endpoints: (a) Pair of cells that maximize the length of the shortest path distances matrix in the dataset. (b) Every cell n, not in the shortest path between already found endpoints is evaluated according to: $s_{\mathcal{E}}(n) := 0.5 \times \min\{S_{kn} + S_{nl} - S_{kl} \,:\, k, l \in \mathcal{E}\}$. S_ij represents the number of nodes in the shortest path between i, j. The new enpoint to be added to the tree structure is the one that maximizes its $s_{\mathcal{E}}(n)$ value respect to the other cells in the dataset.

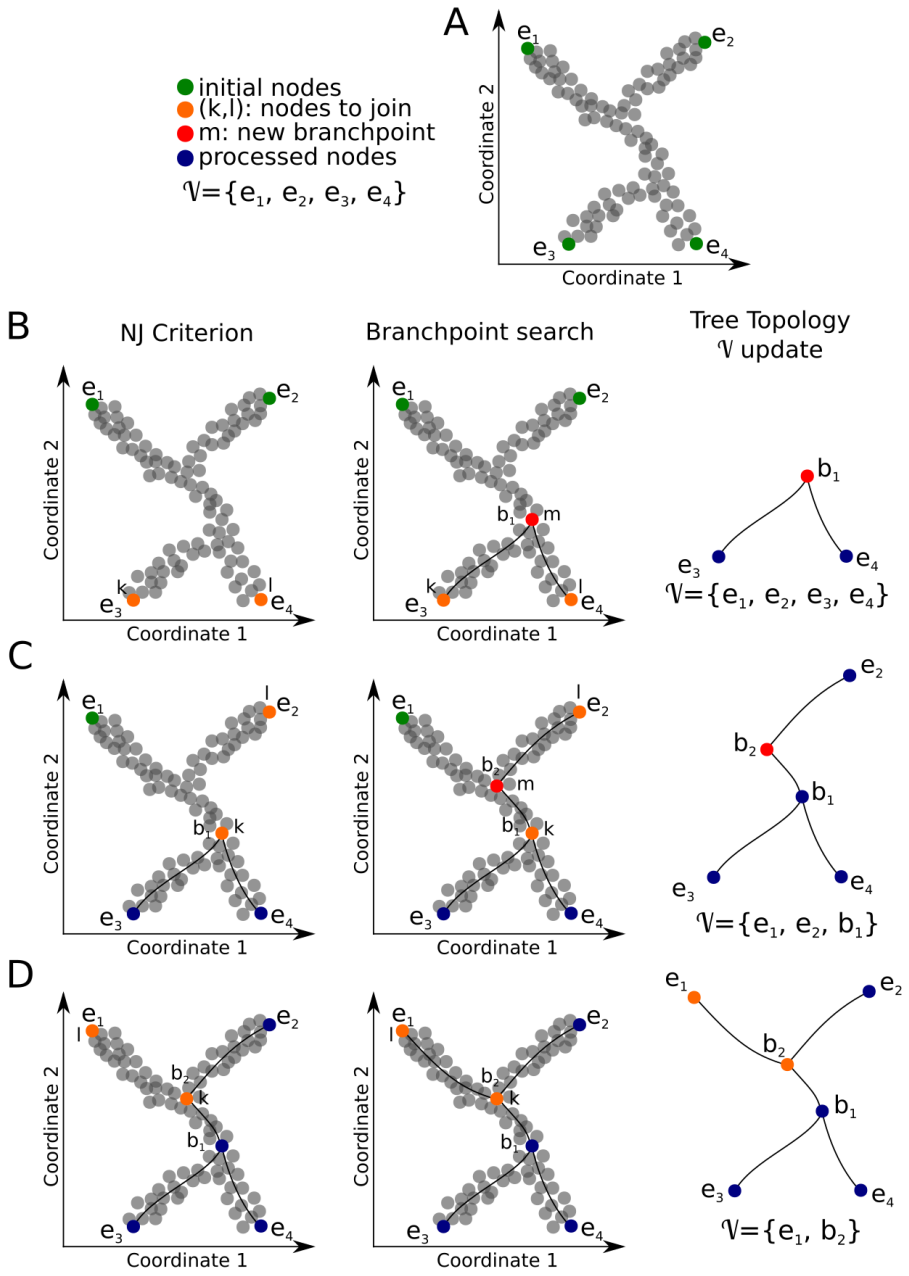

Figure S2: MERLoT's method for finding branchpoints and tree connectivity: The algorithm has two steps (1) Neighbour Joining (NJ) Criterion and (2) branchpoint search. The NJ criterion is applied over the $\mathcal{V}$ vector which is initially set to the set of endpoints (green). In every NJ iteration, two k, l nodes are selected to be joined (orange) for which a branchpoint m is found (red) and the edges between the branchpoint and the joined nodes are added to the topology. The $\mathcal{V}$ vector is updated in every iteration by substracting the k and l nodes and adding the m node. The procedure stops when $|\mathcal{V}|$=2. **(a)** Two-dimensional embedding of cells and initial endpoints used to initialize $\mathcal{V}$. **(b)** First iteration: e3 and e4 and joined through b1. e3-b1 and e4-b1 edges are added to the tree. e3 and e4 are deleted from $\mathcal{V}$ and b1 is added instead. **(c)** Second iteration: e2 and b1 and joined through b2. e2-b2 and b1-b2 edges are added to the tree. e2 and b1 are deleted from $\mathcal{V}$ and b2 is added instead. **(d)** Third iteration: $|\mathcal{V}|$=2, procedure stops and the b2-e1 edge is added to the tree.

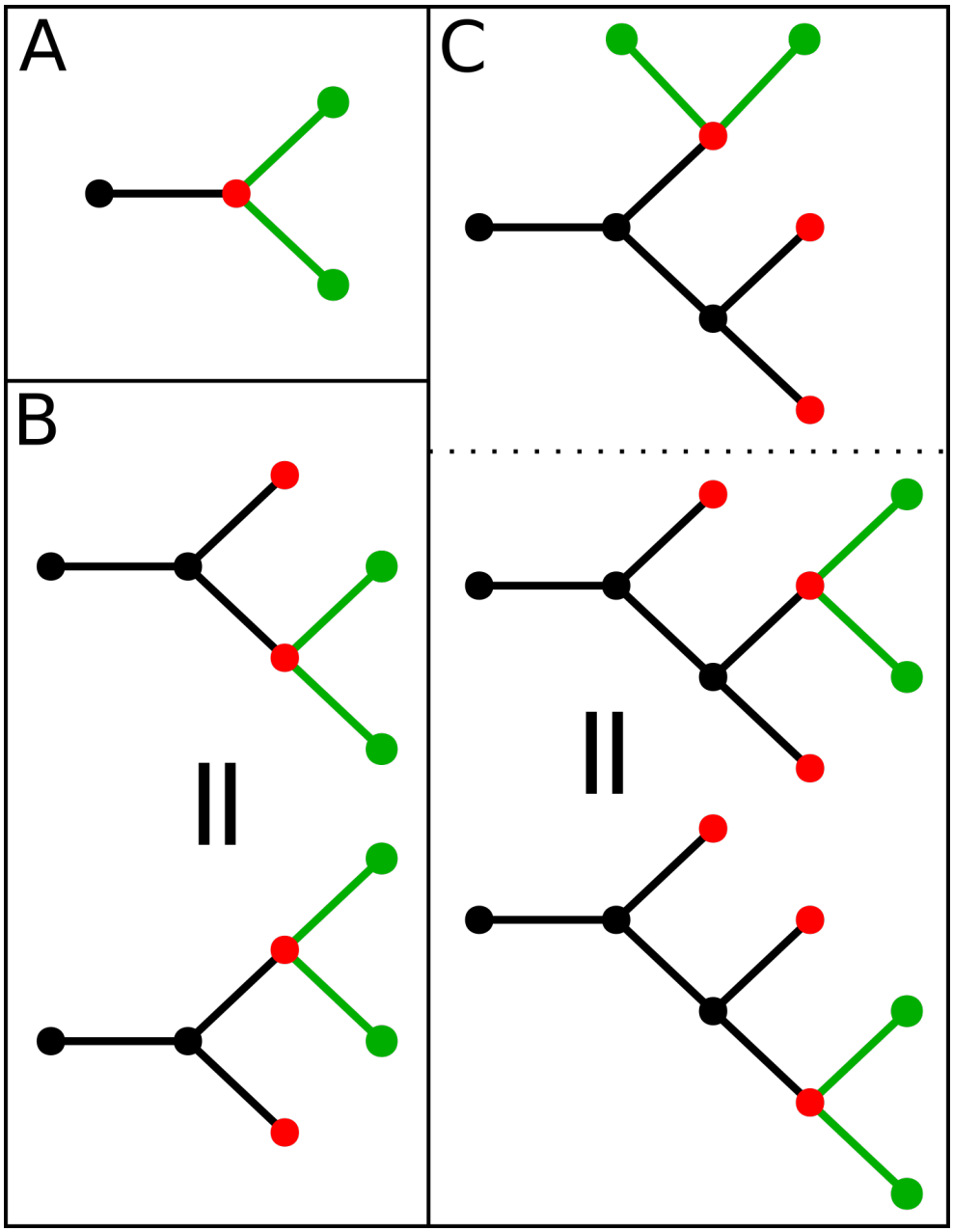

Figure S3: Topology Complexity. **(a)** With one bifurcation only one topology is possible. **(b)** With two bifurcations two topologies are possible. However, they are equivalent in terms of length of the longest subtree. **(c)** More topologies are possible. For the upper case in panel B, three more topologies can be constructed by adding a new bifurcation. Not all topologies are equivalent in terms of the length of the longest sub tree.

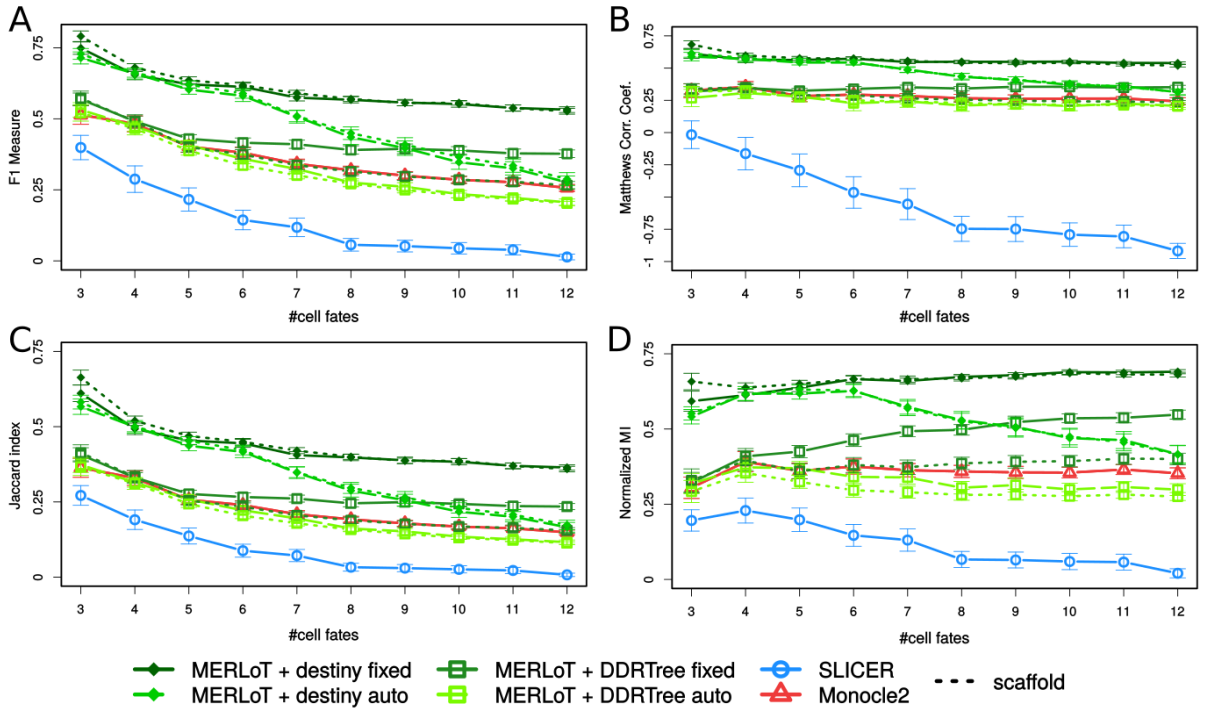

Figure S4: Branch assignment benchmarking. We calculated different scores to assess how good the different methods are for assigning cells to the different measures. **(a)** F1 measure, **(b)** Mathews Correlation Coefficient, **(c)** Jaccard Index and (d) Normalized Mutual Information. For MERLoT, for every flavour, we have included in dotted lines the performance of the Scaffold Tree alone.

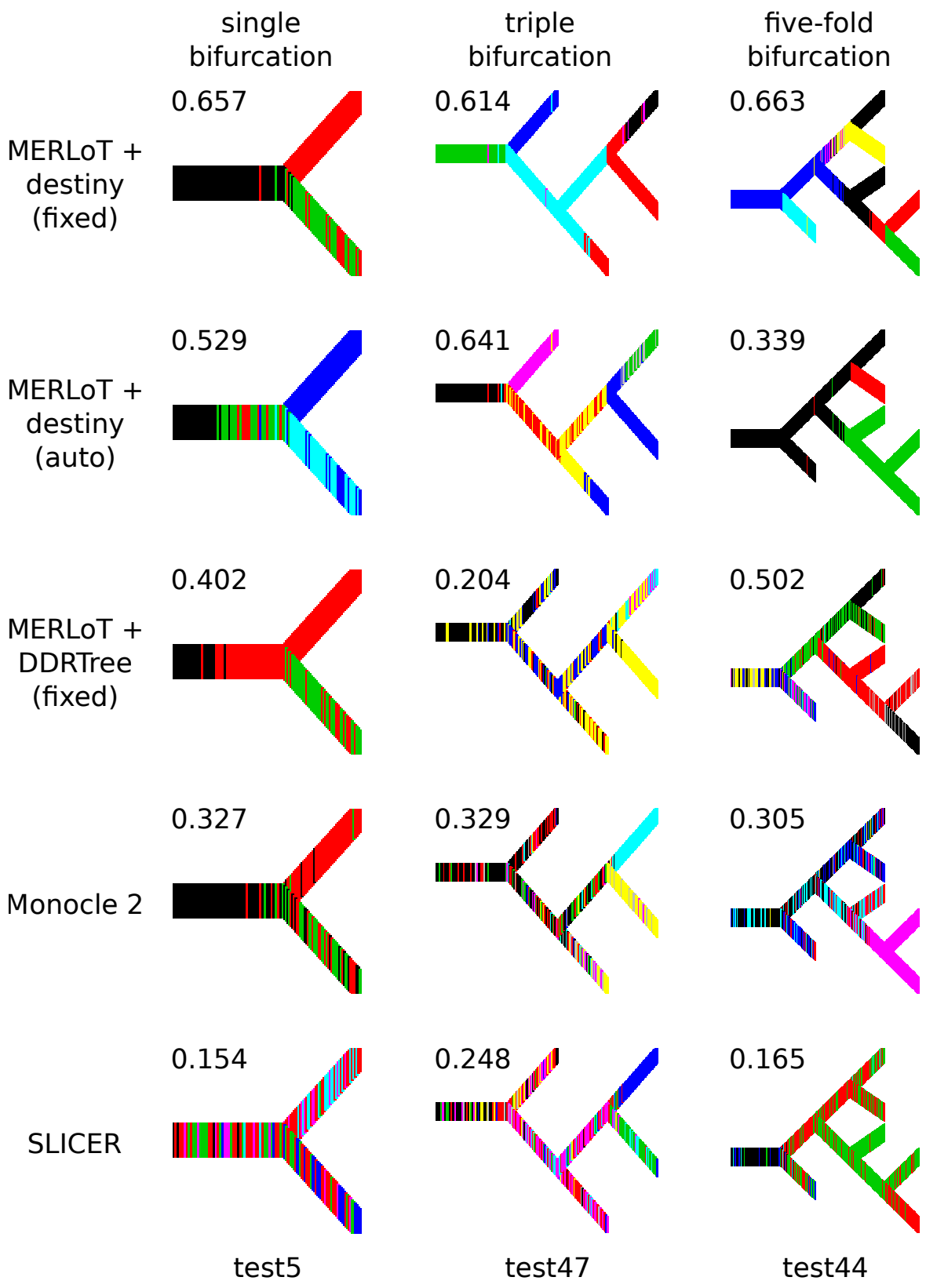

Figure S5: Examples of branch assignments for MERLoT, Monocle2, and SLICER. We show three typical examples from the single (left), triple (middle) and five-fold (right) bifurcation benchmark sets (simulation names at bottom). In each panel we show the lineage tree of the simulated tree, with vertical lines representing the cells, ordered by pseudotime. The color of each line is the branch label predicted for it by the respective method. The NMI of each prediction is in the top left of each panel. We picked examples with NMI values near the average values for each tool and tree topology.

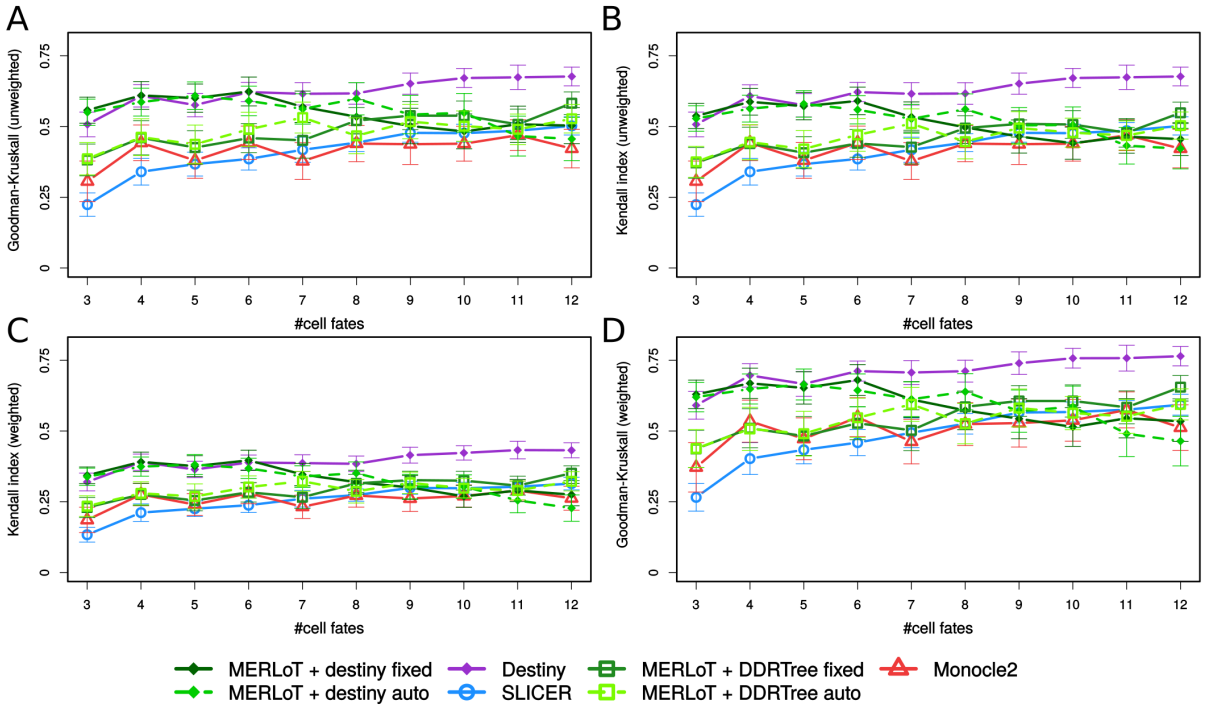

Figure S6: Pseudotime assignment benchmarking. We calculated different scores to assess how good the different methods are for assigning pseudotime values for cells along the longest sub tree in simulations: **(a)** Goodman-Kruskall index (unweighted), **(b)** Kendall index (unweighted), **(c)** Goodman-Kruskall index (weighted) and **(d)** Kendall index (weighted.

Suppl. Note 1: Use of external cluster evaluation measures for branch assignment predictions

As discussed in the main paper, branch assignment predictions can be treated as a clustering problem. A tree reconstruction produces a clustering by assigning cells to branches. This clustering can be compared to the cluster identity of each cell is the branch of the simulated lineage tree it belongs to.

Given a cell c, let $Tr(c)$ be the cluster identity of c in the ground truth and $A(c)$ the cluster assigned to c by the algorithm. We define a pair of two cells c_i, c_j as a...

$$\begin{array}{lll} \text{true positive (TP)} & \text{if} & \mathrm{Tr}(c_i) = \mathrm{Tr}(c_j) \wedge \mathrm{A}(c_i) = \mathrm{A}(c_j) \\ \text{true negative (TN)} & \text{if} & \mathrm{Tr}(c_i) \neq \mathrm{Tr}(c_j) \wedge \mathrm{A}(c_i) \neq \mathrm{A}(c_j) \\ \text{false positive (FP)} & \text{if} & \mathrm{Tr}(c_i) \neq \mathrm{Tr}(c_j) \wedge \mathrm{A}(c_i) = \mathrm{A}(c_j) \\ \text{false negative (FN)} & \text{if} & \mathrm{Tr}(c_i) = \mathrm{Tr}(c_j) \wedge \mathrm{A}(c_i) \neq \mathrm{A}(c_j) \end{array}$$

Using these four values, many popular clustering indices can be computed (Fig S4): the F1 measure (Fig. S4A) is $2PR/(P+R)$, (where P is the precision $TP/(TP+FP)$ and R is the recall $TP/(TP+FN)$), the Matthews Correlation Coefficient (MCC) (Fig. S4B) is

$$\frac{(TP \cdot TN - FP \cdot FN)}{\sqrt{((TP+FP)(TP+FN)(TN+FP)(TN+FN))}},$$

and the Jaccard Index (Fig. S4C) is $TP/(TP+FP+FN)$

While all the aforementioned indices and measures are well established, they are suboptimal performance indicators for the problem at hand, since they don't take cluster structure into consideration. As the simulated lineage trees become bigger and more complex, the number of cell pairs that are not in the same tree segment is going to grow much faster than the number of cell pairs in different tree segments. The number of TPs will grow much slower than the number of TNs, something that will, for example, inflate the the MCC index. Additionally, the number of possible FNs becomes much higher with each additional segment added to the tree, something that affects the Jaccard index and the F1 score. In short, these measures don't take into account the number of clusters in the data, and as such are suboptimal descriptors of clustering performance in comparison to NMI.