Reversed graph embedding resolves complex single-cell trajectories

Xiaojie Qiu[1,2], Qi Mao[3], Ying Tang[4], Li Wang[5], Raghav Chawla[2], Hannah A Pliner[2] & Cole Trapnell[1,2]

Single-cell trajectories can unveil how gene regulation governs cell fate decisions. However, learning the structure of complex trajectories with multiple branches remains a challenging computational problem. We present Monocle 2, an algorithm that uses reversed graph embedding to describe multiple fate decisions in a fully unsupervised manner. We applied Monocle 2 to two studies of blood development and found that mutations in the genes encoding key lineage transcription factors divert cells to alternative fates.

Most cell-state transitions, whether in development, reprogramming, or disease, are characterized by cascades of gene expression changes. We recently introduced a technique called 'pseudotemporal ordering', which applies machine learning to single-cell transcriptome sequencing (scRNA-seq) data to order cells along a reconstructed 'trajectory' of differentiation or other type of inferred biological transition[1]. Despite intense efforts to develop scalable, accurate pseudotime reconstruction algorithms (recently reviewed in ref. 2), state-of-the-art tools have several major limitations. Most of the methods can only reconstruct linear trajectories, whereas others, such as Wishbone[3] or DPT[4], support branch identification with heuristic procedures; however, they are either restricted to identifying a single branch or require the user to specify the number of branches and cell fates (which may not be known) as a parameter.

Here we describe Monocle 2 (**Supplementary Software** and https://github.com/cole-trapnell-lab/monocle-release), which applies reversed graph embedding (RGE)[5,6], a recently developed machine-learning strategy, to accurately reconstruct complex single-cell trajectories. Monocle 2 requires no a priori information about the genes that characterize the biological process, the number of cell fates or branch points in the trajectory, or the design of the experiment. Monocle 2 produces more accurate, robust trajectories than its previous version, as well as more recently developed methods.

Monocle 2 begins by identifying genes that define a biological process using an unsupervised procedure we term 'dpFeature'. The procedure works by selecting the genes that are differentially expressed between clusters of cells identified with *t*-distributed stochastic neighbor embedding (tSNE) dimension reduction followed by density-peak clustering[7]. When applied to four different data sets[1,8–10] most of the genes returned by dpFeature were also recovered by a semisupervised selection method guided by aspects of the experimental design and were highly enriched for relevant Gene Ontology terms, confirming that dpFeature is a powerful and general approach for unsupervised feature selection (**Supplementary Figs. 1–3**).

To develop a pseudotime-trajectory-reconstruction algorithm that does not require cell fate or branch numbers as input, we used RGE[5,6], a machine-learning technique to learn a parsimonious 'principal graph'. Informally, a principal graph is like a principal curve[11] that passes through the 'middle' of a data set but is allowed to have branches[12]. However, learning a principal graph that describes a population of scRNA-seq profiles is very challenging because each expressed gene adds an additional dimension to the gene expression space, and learning geometry is dramatically harder in high-dimensional spaces[13]. RGE solves this problem by finding a mapping between the high dimensional gene expression space and a much lower dimensional space while simultaneously learning the structure of the graph in this reduced space.

By default, Monocle 2 uses DDRTree[5,6], a scalable RGE algorithm, to learn a principal tree on a population of single cells. The tree is intended to describe changes to global gene expression as a cell progresses through the biological process under study (**Fig. 1a**). In contrast to other methods[1,3,4,14], Monocle 2 identifies branch points that describe significant divergences in cellular state automatically. Monocle 2 is also equipped with alternative RGE methods[5,6], including one that in principle can learn cyclical or disjoint trajectories, although doing so requires some degree of parameter optimization on behalf of the user.

To assess the accuracy of Monocle 2, we applied it to myoblasts, which we previously reported differentiate along a linear trajectory[1] (**Fig. 1b**). Unexpectedly, Monocle 2 was able to reconstruct a trajectory with a single branch point that led to two outcomes (**Fig. 1c**). The same genes selected with dpFeature (**Supplementary Fig. 1** and Online Methods) were used for pseudotime ordering for both of Monocle 1 and Monocle 2. Some genes associated with mitogen withdrawal, such as *CCNB2*, showed similar kinetics on both branches; however, a number of genes required for muscle contraction were strongly activated on only one branch (**Supplementary Fig. 4**). A global search for genes with significant branch-dependent expression using branch expression analysis modeling (BEAM)[15] revealed that cells along these two outcomes, F_1 and F_2, differed in the

[1]Molecular and Cellular Biology Program, University of Washington, Seattle, Washington, USA. [2]Department of Genome Sciences, University of Washington, Seattle, Washington, USA. [3]HERE Company, Chicago, Illinois, USA. [4]Department of Physics and Astronomy, Shanghai Jiao Tong University, Shanghai, China. [5]Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, USA. Correspondence should be addressed to C.T. (coletrap@uw.edu).

RECEIVED 22 FEBRUARY; ACCEPTED 20 JULY; PUBLISHED ONLINE 21 AUGUST 2017; DOI:10.1038/NMETH.4402

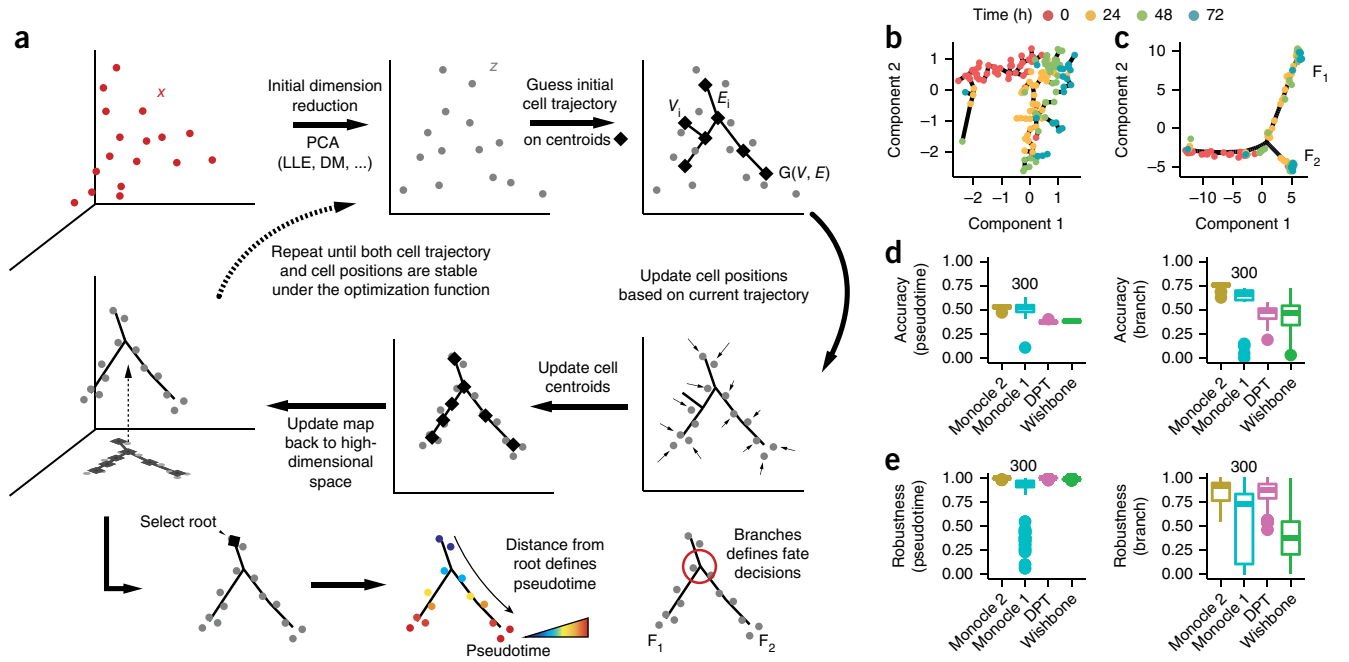

Figure 1 | Monocle 2 discovers a cryptic alternative outcome in myoblast differentiation. (**a**) Monocle 2 automatically learns single-cell trajectories and branch points by reversed graph embedding (Online Methods). Each cell is represented as a point in high-dimensional space (*x*), where each dimension corresponds to the expression level of an ordering gene. Data are projected onto a lower-dimensional space (*z*) by dimension-reduction methods such as PCA, and Monocle 2 constructs a spanning tree on a set of centroids (diamonds) chosen automatically using *k*-means clustering. Cells are then shifted toward the nearest tree vertex, vertex positions are updated to 'fit' cells, a new spanning tree is learned, and the process is iterated until the tree and cells converge. The user then selects a tip as the 'root', each cell's pseudotime is calculated as its geodesic distance along the tree to the root, and its branch is automatically assigned based on the principal graph. (**b**) Differentiating human skeletal myoblasts projected onto the first two components from an ICA by Monocle 1. Black segments indicate cells connected in a minimum spanning tree. (**c**) Cells from **b** projected into a two-dimensional space by Monocle 2 using DDRTree. Black segments indicate the graph learned as illustrated in **a**. The components are distinct from those shown in **b**. Trajectory outcomes are indicated as F_1 and F_2. (**d**,**e**) Accuracy (**d**) and consistency (**e**) of pseudotime calculation (left) or branch assignments (right) from each algorithm under repeated subsampling of 80% of the cells on the Paul *et al.* data set[10]. For **d**, a marker-based ordering scheme was used as ground truth (Online Methods). For **e**, all pairwise downsamplings were used to calculate the Pearson's rho and adjusted Rand index (ARI). For benchmarking, Monocle 2, DPT, and Wishbone used the full data set, whereas Monocle 1 used only a random downsample of 300 cells.

expression of 887 genes (FDR < 10%), including those encoding numerous components of the contractile muscle program. BEAM analysis suggested that only F_1 represented successful progression to fused myotubes (**Supplementary Fig. 4**), which was consistent with previous immunofluorescence measurements of *MYH2* expression showing that a substantial fraction of isolated nuclei that lack MYH2 are not incorporated into myotubes (Figs. 1 and 4 in Trapnell *et al.*[1]).

We used a simulation of differentiation controlled by a hypothetical gene regulatory network[16] (modeled by a set of stochastic differential equations) to demonstrate that Monocle 2 robustly and accurately reconstructed trajectories with up to three fates (**Supplementary Figs. 5–8** and **Supplementary Data 1** and **2**)[17]. In contrast to other methods, Monocle 2 also accurately learned a complex tree with five branches in a fully automatic fashion (**Supplementary Fig. 6b** and **Supplementary Data 3**).

We compared Monocle 2 to other single-cell trajectory inference methods, including Monocle 1 (ref. 1), Wishbone[3], Diffusion Pseudotime (DPT)[4], and SLICER[14]. Unlike Monocle 2, these methods do not construct an explicit tree. Instead, they order cells on the basis of pairwise geodesic distances between them as approximated by a nearest-neighbor graph (Wishbone and SLICER) or a minimum spanning tree (Monocle 1), or are calculated analytically (DPT). Wishbone, SLICER, and DPT identify branches implicitly by analyzing patterns in the pseudotime orderings that are inconsistent with a linear trajectory. Furthermore, Wishbone assumes that the trajectory has exactly one branch point, whereas DPT can detect more than one, but it provides no means of automatically determining how many genuine branches exist in the data. We hypothesized that Monocle 2's explicit trajectory structure would yield more robust pseudotimes and branch assignments than alternative algorithms.

We tested each algorithm using data from Paul *et al.*[10], who analyzed transcriptomes of several thousands of differentiating blood cells[10]. Monocle 2, DPT, and Wishbone produced qualitatively similar trajectories, with common myeloid progenitor (CMP) cells residing upstream of a branch at which granulocyte and monocyte progenitor (GMP) and erythroid cells diverge (**Supplementary Figs. 9–11**). These algorithms produced orderings that were highly correlated with a 'reference ordering', which was constructed using a panel of markers similar to the approach introduced by Tirosh *et al.*[18], whereas SLICER and Monocle 1 orderings were less correlated. Monocle 2 assigned cells to branches as accurately or more so than the other methods (**Fig. 1d** and **Supplementary Fig. 10**), but Monocle 2's assignments were far more consistent when provided with subsampled fractions of the cells (**Fig. 1e** and **Supplementary Fig. 9f,g**). When DPT was used on the myoblast data, it positioned the most fully differentiated cells along a major branch, and the incompletely

differentiated cells were split along a minor branch or not assigned to either; Wishbone failed to discriminate correctly between the two outcomes (**Supplementary Fig. 12**). Although Monocle 2 can be tuned for several user-specified parameters, results on the four data sets were similar to the default parameters over widely varying parameter values (**Supplementary Figs. 13** and **14**). Monocle 2's running time scaled linearly in the number of input cells, consistent with its linear algorithmic complexity, and processed 8,365 cells in 9 min (**Supplementary Fig. 13c**). These benchmarks demonstrates that Monocle 2 produces trajectories that are as accurate and more robust than state-of-art methods and yet makes fewer assumptions regarding the number of cell fates generated by the trajectory.

We also assessed alternative algorithms offered in Monocle 2 for dimensionality reduction and graph learning. DDRTree, SimplePPT, and SGL-tree, which implement RGE to learn principal trees, reported highly concordant trajectories when the data was initially reduced with principal component analysis (PCA), independent component analysis (ICA), and diffusion maps (**Supplementary Fig. 15**). Locally linear embedding (LLE), a reduction technique known to be highly sensitive to tuning parameters, sometimes led to incorrect reconstructions with SimplePPT. L1-graph, an RGE algorithm that can learn graphs with multiple components or cycles, often reported less refined graphs with numerous minor branches, but it captured the overall trajectory structure accurately.

Monocle 2 can, in principle, learn complex trajectories with many branches, but such processes might not be well described by a two-dimensional projection of the data. We thus reanalyzed the data from Paul *et al.*[10] in ten dimensions (selected on the basis of variance explained by principle components) rather than in the default of two. This higher-dimensional trajectory contained five branching events that led to six different outcomes, with the cells classified by Paul *et al.*[10] (fully differentiated monocytes, neutrophils, eosinophil, basophils, dendritic cells, megakaryocytes, and erythrocytes) confined to distinct outcomes (**Supplementary Fig. 16**). Our results confirmed that Monocle 2 can resolve complex branching processes.

Although the trajectories determined by Monocle 2 for differentiating myoblasts and common myeloid progenitors were broadly consistent with the known sequence of regulatory events governing those processes, we sought further experimental means of validating the structure of the algorithm's trajectories. Lesions in the genes encoding key developmental regulators frequently lead to abnormal phenotypes and dysregulated gene expression programs; we thus hypothesized that a trajectory from mutant mice might look very different from that of the wild-type (WT) mice. Recently, Olsson *et al.*[9] profiled several hundred FACS-sorted cells during various stages of mouse myelopoiesis, i.e., LSK, CMP, GMP and LKCD34$^+$ cells. Monocle 2 reconstructed a trajectory from these cells with two major branches and three distinct fates (**Fig. 2** and **Supplementary Figs. 17** and **18**). Lin$^-$Sca1$^+$c-Kit$^+$ (LSK) stem and multipotent progenitor cells were concentrated at one tip of the tree, which we designated the root, whereas the CMP, GMP, and LKCD34$^+$ cells were distributed over the remainder of the tree (**Fig. 2a** and **Supplementary Fig. 18a**).

Monocle 2 placed the cells classified by Olsson *et al.*[9] as erythrocytes or megakaryocytes on a path to outcome F_E, whereas granulocytes and monocytes were confined to outcomes F_G and F_M, respectively. Genes that were associated with the granulocytic and monocytic programs became progressively more differentially expressed following the second branch (**Supplementary Fig. 17b,c**). Many genes with significant branch-dependent expression

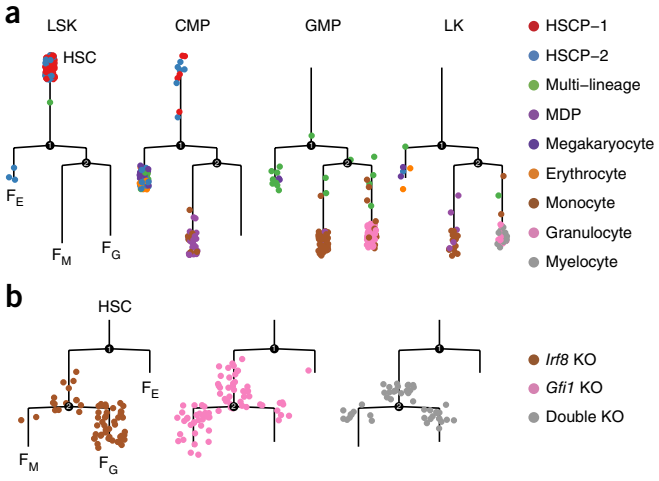

Figure 2 | Monocle 2 trajectories reveal that genetic perturbations divert cells to alternative outcomes. (**a**) Monocle 2 trajectory of differentiating blood cells, as collected by Olsson *et al.*[9]. Each subpanel corresponds to cells collected via fluorescence-activated cell sorting (FACS), including stem–multipotent progenitor cells (LSK; Lin$^-$Sca1$^+$c-Kit$^+$), common myeloid progenitor (CMP) cells, granulocyte and monocyte progenitor (GMP) cells, and LK cells (Lin$^-$c-Kit$^+$CD34$^+$) that included granulocytic precursors. Cells are colored according to their classification by the authors of the original study[9]. HSCP-1 and HSCP-2 indicate human stem and progenitor cells replicates 1 and 2, respectively. (**b**) Cells with an *Irf8* or *Gfi1* knockout (KO) were diverted into the alternative granulocyte or monocyte branch, respectively. Double-knockout cells were localized to both the granulocyte and monocyte branches, but were concentrated near the branch point. Two branch points were identified, one that divided the erythroid or megakaryocyte outcome (F_E) from the GMP cells, which then branched to the monocyte (F_M) and granulocyte (F_G) outcomes. Trajectories were reconstructed in four dimensions but are rendered in two dimensions.

(BEAM test[15], FDR < 1%) were bound at their promoters by Irf8 or Gfi1, key activators of the monocytic and granulocytic expression programs, respectively (**Supplementary Fig. 17c,d**).

When we provided the cell data from mice lacking *Gfi1* or *Irf8* that were collected by Olsson *et al.*[9] to Monocle 2, there was no substantial alteration to the structure of the myeloid differentiation trajectory (**Fig. 2b**). However, cells from *Gfi1*$^{-/-}$ mice were largely excluded from the branch that was occupied by granulocytes from the WT mice, and cells from the *Irf8*$^{-/-}$ mice were depleted from the monocyte branch in the WT mice. Thus, the loss of a gene known to activate a fate-specific expression program seemed to divert cells to the opposite fate. Cells from *Gfi1*$^{-/-}$*Irf8*$^{-/-}$ double-knockout mice were present on both the monocyte and granulocyte branches, but they were concentrated closer to the branch point and away from the tips of the tree, suggesting that these cells were not fully differentiated (**Supplementary Fig. 18c**).

The *Gfi1*$^{-/-}$ cells on the branch to outcome F_M expressed higher levels of the genes normally associated with granulocytes than those associated with WT monocytes (**Supplementary Fig. 18** and Online Methods). Similarly, cells from *Irf8*$^{-/-}$ mice on the branch to outcome F_G showed aberrantly high levels of monocyte -specific genes. Thus, whereas *Gfi1* and *Irf8* are required for generating normal granulocytes and monocytes, other regulators must contribute to activating the specific programs of these cell types. An analysis of genetic perturbations from the large-scale transcriptomic study of hematopoiesis reported by Paul *et al.*[10] also revealed diversions of cells onto specific

branches of the trajectory (**Supplementary Fig. 19g**,**h**), suggesting that the loss of a key fate regulator can divert cells from one fate to another, and that Monocle 2 and BEAM analysis can be used to dissect the genetic architecture controlling cell fate decisions (**Supplementary Figs. 19f**,**g** and **20**).

scRNA-seq has spurred an explosion of computational methods to infer the precise sequence of gene regulatory events that drive transitions from one cellular state to another. However, most of the methods rely on strong assumptions about the structure of a biological trajectory. Many also require the user to supervise trajectory inference, inject large amounts of a priori biological knowledge, or both.

Monocle 2 learns complex cellular trajectories with multiple branches in a fully data-driven, unsupervised fashion with only limited assumptions regarding its structure. In contrast to previous methods that infer branch structure using heuristic analyses of pairwise distances between cells, Monocle 2 can use the principal graph that it learns to directly identify developmental fate decisions. We have demonstrated through extensive benchmarking that Monocle 2 compares favorably with other tools, such as Wishbone, without requiring the user to specify the structure of the trajectory.

Previously, we showed that loss of interferon signaling can create a new branch in an otherwise linear trajectory that reflects the response of dendritic cells to antigen[15]. Here we show that cells from mice that lack transcription factors required for establishing specific myeloid fates were diverted onto alternative fates of the same trajectory without altering its structure. Why some loss-of-function mutations create branches, whereas others divert cells along existing ones, is unclear, but this question underscores the power of analyzing single-cell trajectories. We anticipate that Monocle 2 will also be useful for single-cell chromatin accessibility[19] or three-dimensional structure[20] analysis and are confident that it will help reveal how various layers of gene regulation coordinate developmental decision making within individual cells.

METHODS

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

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

We thank I. Tirosh for discussions on marker-based ordering, F. Theis and F.A. Wolf for discussions on the data analysis with DPT from Paul *et al.*[9], and members of the Trapnell laboratory for comments on the manuscript. This work was supported by US National Institutes of Health (NIH) grants DP2 HD088158 (C.T.) and U54 DK107979 (C.T.); C.T. is partly supported by a Dale. F. Frey Award for Breakthrough Scientists and an Alfred P. Sloan Foundation Research Fellowship; and H.A.P. is supported by a National Science Foundation (NSF) Graduate Research Fellowship (DGE-1256082).

AUTHOR CONTRIBUTIONS

X.Q., Q.M., and C.T. designed and implemented Monocle 2; X.Q. performed the analysis; Y.T. and L.W. contributed to the technical design; R.C. and H.A.P. performed the testing; C.T. conceived the project; and all authors wrote the manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

1. Trapnell, C. *et al. Nat. Biotechnol.* **32**, 381–386 (2014).
2. Kumar, P., Tan, Y. & Cahan, P. *Development* **144**, 17–32 (2017).
3. Setty, M. *et al. Nat. Biotechnol.* **34**, 637–645 (2016).
4. Haghverdi, L., Büttner, M., Wolf, F.A., Buettner, F. & Theis, F.J. *Nat. Methods* **13**, 845–848 (2016).
5. Mao, Q., Wang, L., Goodison, S. & Sun, Y. in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 765–774 (ACM, 2015).
6. Mao, Q., Wang, L., Tsang, I. & Sun, Y. *IEEE Trans. Pattern Anal. Mach. Intell.* https://doi.org/10.1109/TPAMI.2016.2635657 (2016).
7. Rodriguez, A. & Laio, A. *Science* **344**, 1492–1496 (2014).
8. Treutlein, B. *et al. Nature* **509**, 371–375 (2014).
9. Olsson, A. *et al. Nature* **537**, 698–702 (2016).
10. Paul, F. *et al. Cell* **163**, 1663–1677 (2015).
11. Hastie, T. & Stuetzle, W. *J. Am. Stat. Assoc.* **84**, 502–516 (1989).
12. Gorban, A.N. & Zinovyev, A.Y. in *Handbook of Research on Machine-learning Applications and Trends: Algorithms, Methods, and Techniques* 28–59 (Information Science Reference, Hershey, Pennsylvania, USA, 2009).
13. Bellman, R. *The Theory of Dynamic Programming* (DTIC Document, 1954).
14. Welch, J.D., Hartemink, A.J. & Prins, J.F. *Genome Biol.* **17**, 106 (2016).
15. Qiu, X. *et al. Nat. Methods* **14**, 309–314 (2017).
16. Qiu, X., Ding, S. & Shi, T. *PLoS One* **7**, e49271 (2012).
17. Tang, Y., Yuan, R., Wang, G., Zhu, X. & Ao, P. *arXiv:1611.07140* (2016).
18. Tirosh, I. *et al. Nature* **539**, 309–313 (2016).
19. Cusanovich, D.A. *et al. Science* **348**, 910–914 (2015).
20. Ramani, V. *et al. Nat. Methods* **14**, 263–266 (2017).

ONLINE METHODS

Reversed graph embedding. Monocle 2 uses a technique called reversed graph embedding[5,6,21] (RGE) to learn a graph structure that describes a single-cell experiment. RGE simultaneously learns a principal graph that represents the cell trajectory, as well as a function that maps points on the trajectory (which is embedded in low dimensions) back to the original high-dimensional space. RGE aims to learn both a set of latent points $\mathcal{Z} = \{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_N\}$, where N is the number of the set (or cell numbers) and an undirected graph $\mathcal{G}$ that connects these latent points. The latent points $\mathcal{Z}$ in the low-dimensional space corresponds to the input data $\mathcal{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ in the high-dimensional space. The graph $\mathcal{G} = (V, \varepsilon)$ contains a set of vertexes $V = \{V_1,\ldots,V_N\}$ and a set of weighted, undirected edges ε, where each V_i corresponds to latent point $\boldsymbol{z}_i$, so the graph also resides in the latent, low-dimensional space.

In the context of the single-cell trajectory construction problem, $\boldsymbol{x}_i$ is typically a vector of the feature genes' expression values (for example, based on dpFeature selection, see **Supplementary Note**) of the ith cell in a scRNA-seq experiment, $\mathcal{G}$ is the learned trajectory (for example, a tree) along which the cells transit, and $\boldsymbol{z}_i$ is the principal point on $\mathcal{G}$ corresponding to the cell $\boldsymbol{x}_i$.

RGE learns the graph $\mathcal{G}$, as well as a function that maps back to the input data space. We let b_{ij} denote the weight of edge (V_i, V_j), which represents the connectivity between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$. In other words, $b_{ij} > 0$ means that edge (V_i, V_j) exists in $\mathcal{G}$, and 0 otherwise. We define $f_{\mathcal{G}}$ as the projection function from $\boldsymbol{z}_i$ to some point in the high-dimensional space. To learn $\mathcal{G}$, $\mathcal{Z}$ and $f_{\mathcal{G}}$, we need to optimize

$$\min_{\mathcal{G}\in G_b} \min_{f_{\mathcal{G}}\in\mathcal{F}} \min_{\mathcal{Z}} \sum_{(V_i,V_j)\in\varepsilon} b_{i,j} \| f_{\mathcal{G}}(\boldsymbol{z}_i) - f_{\mathcal{G}}(\boldsymbol{z}_j) \|^2 \quad (1)$$

where G_b is a set of feasible graph structures with parameters $\{b_{i,j}, \forall i,j\}$, and $\mathcal{F}$ is a set of functions that map a latent, low-dimensional point to a point in the original, high-dimensional space.

As shown previously[5], the above-mentioned optimization procedure will learn graph structures in the latent space, but it does not measure the deviations of the latent points to the observed data, i.e., no effort is made to ensure that the graph nodes are embedded in a way that is relevant to the cloud of observed data points. To learn a graph that describes the overall structure of the observed data, RGE aims to position the latent points such that their image under the function $f_{\mathcal{G}}$ (i.e., their corresponding positions in the high-dimensional space) will be close to the input data while also ensuring that neighbor points on the low-dimensional principal graph are 'neighbors' in the input dimension. The optimization problem is formulated as

$$\min_{\mathcal{G}\in G_b} \min_{f_{\mathcal{G}}\in\mathcal{F}} \min_{\mathcal{Z}} \sum_{i=1}^{N} \| \boldsymbol{x}_i - f_{\mathcal{G}}(\boldsymbol{z}_i) \|^2 + \frac{\lambda}{2} \sum_{(V_i,V_j)\in\varepsilon} b_{i,j} \| f_{\mathcal{G}}(\boldsymbol{z}_i) - f_{\mathcal{G}}(\boldsymbol{z}_j) \|^2 \quad (2)$$

where λ is a parameter that adjusts the relative strength of these two summations.

In practice, implementing reversed graph embedding requires that we place some constraints on G_b and $f_{\mathcal{G}}$, as summarized briefly in the following sections describing DDRTree, SimplePPT, and SGL-tree. These are different ways to implement the reversed graph embedding framework outlined above. Although Monocle 2 uses DDRTree as the default trajectory reconstruction algorithm, in cases where the data under study only includes important low-dimensional features (for example, in the cases where important genes or proteins are measured in the single-cell RT–PCR or CyTOF experiments), SimplePPT or SGL-tree can be used directly to reconstruct the trajectory in the same dimensions of the data. Each of the three RGE methods can be used in conjunction with various algorithms to reduce the dimensionality of the data (for example, PCA, diffusion maps, etc.). For example, users could project their data onto the top three diffusion components and then learn a trajectory in this space using SimplePPT.

SimplePPT: a simple algorithm for principal trees. SimplePPT is the first RGE technique proposed for learning a tree structure to describe a set of observed data points. The tree can be learned in the original space or in some lower dimension retrieved by dimensionality reduction methods such as PCA[21]. SimplePPT makes some choices that simplify the optimization problem. Notably, $f_{\mathcal{G}}(\boldsymbol{z}_i)$ is optimized as one single variable instead of two separate sets of variables. Moreover, the loss function in the RGE is replaced by the empirical quantization error, which serves as the measurement between the $f_{\mathcal{G}}(\boldsymbol{z}_i)$ and its corresponding observed points $\boldsymbol{x}_i$. The joint optimization of $f_{\mathcal{G}}(\boldsymbol{z}_i)$ is efficient from the perspective of optimization with respect to $\{b_{ij}\}$, which is solved by simply finding the minimum spanning tree.

The principal $\mathcal{L}_1$ graph algorithm. Mao *et al.*[6] later proposed an extension of SimplePPT that can learn arbitrary graphs rather than just trees, which describes large data sets embedded in the same space as the input. An $\mathcal{L}_1$ graph is a sparse graph that is based on the assumption that each data point (or cell) has a small number of neighborhoods in which the minimum number of points that span a low-dimensional affine subspace[22] passing through that point. In addition, there may be noise in certain elements of $\boldsymbol{z}_i$, and a natural idea is to estimate the edge weights by tolerating these errors. In general, a sparse solution is more robust and facilitates the consequent identification of test sample (or sequenced single-cell samples). Unlike SimplePPT, this method learns the graph by formulating the optimization as a linear programming problem.

In the same work[6], the authors also proposed a generalization of SimplePPT, which we term SGL-tree (Principal Graph and Structure Learning for tree), to learn tree structure for a large data set by considering the clustering of data points similarly to that in DDRTree.

DDRTree: discriminative dimensionality reduction via learning a tree. DDRTree[5], the default RGE technique used by Monocle 2, provides two key features that are not offered by the SimplePPT learning framework. First, DDRTree does not assume that the graph resides in the input space, and it can reduce its dimensionality while learning the trajectory. Second, it also does not require that there be one node in the graph per data point, which greatly accelerates the algorithm and reduces its memory footprint.

Like SimplePPT, DDRTree learns a latent point for each cell, along with a linear projection function $f_{\mathcal{G}}(\boldsymbol{z}_i) = \boldsymbol{W}\boldsymbol{z}_i$, where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_d] \in R^{D\times d}$ is a matrix with columns that form an

orthogonal basis $\{w_1, \ldots, w_d\}$ (D is the dimension of feature genes, and d is the dimension of latent space). DDRTree simultaneously learns a graph on a second set of latent points $\mathcal{Y} = \{y_k\}_{k=1}^{K}$. These points are treated as the centroids of $\{z_i\}_{i=1}^{N}$ where $K \leq N$ and the principal graph is the spanning tree of those centroids. The DDRTree scheme works by optimizing

$$\min_{\boldsymbol{W},\boldsymbol{B},\boldsymbol{R},\mathcal{Y},\mathcal{Z}} \sum_{i=1}^{N} \| x_i - \boldsymbol{W} z_i \|^2 + \frac{\lambda}{2} \sum_{k,k'} b_{k,k'} \| \boldsymbol{W} y_k - \boldsymbol{W} y_{k'} \|^2 + \gamma \left[\sum_{k=1}^{K} \sum_{i=1}^{N} r_{i,k} (\| z_i - y_k \|^2 + \sigma \log r_{i,k}) \right] \quad (3)$$

s.t. $\{b_{i,j}\}$ is a spanning tree

$$\boldsymbol{W}^T \boldsymbol{W} = I$$

$$\sum_{k=1}^{K} r_{i,k} = I, r_{i,k} \geq 0, \forall i,k$$

In effect, the algorithm acts as soft K-means clustering on points $\mathcal{Z}$, and it jointly learns a graph on the K-cluster centers. The matrix $\boldsymbol{R}$ with the (i, k)th element as $r_{i,k}$ transforms the hard assignments used in K-means into soft assignments with $\sigma > 0$ as a regularization parameter for soft clustering. The above-mentioned problem contains a number of analytical steps and can be solved by alternating optimization until convergence. Moreover, because some of the more computationally expensive numerical operations involve matrices that are K dimensional (instead of N dimensional); they have complexity that is invariant of the size of the input data for a small fixed K. In Monocle 2, we provide a procedure to automatically choose a value of K that should work well for a wide range of data sets based on the number of cells N in the experiment:

$$K = \begin{cases} N, & \text{if } N < 100, \\ \dfrac{2 \times 100 \times \log(N)}{\log(N) + \log(100)} & \text{otherwise} \end{cases} \quad (4)$$

During the first optimization iteration, these K centroids are initialized by using k-means clustering in the low-dimensional space.

Pseudotime calculation and branch assignment. By default, Monocle 2 calls DDRTree to learn the principal tree that describes a single-cell experiment and then projects each cell onto its nearest location on the tree. Monocle 2 allows users to conveniently select a tip of the tree as the root, and then it transverses the tree from the root, computing the geodesic distance of each cell to the root cell, which is taken as its pseudotime, while assigning branch or segment simultaneously.

DDRTree returns a principal tree of the centroids of cell clusters in low dimension. To calculate pseudotimes, Monocle 2 projects the cells' latent points $\mathcal{Z}$, to the principal graph formed by principal points, $\mathcal{Y}$. For each latent point not near 'tip' principal points (i.e., end nodes of the principal tree), Monocle 2 finds the nearest line segment on the principal tree and then projects it to the nearest point on that segment. More formally, we can define a vector between a cell $c = (c_1, c_2, \ldots)$, where $c_1, c_2, \ldots$ denote the coordinates of the cell in the latent space, and the nearest principal point A by $\overrightarrow{Ac}$. The line segment formed by the two nearest principal points ($A = (A_1, A_2, \ldots)$ and $B = (B_1, B_2, \ldots)$) is $\overrightarrow{AB}$. We then calculate t as

$$t = \frac{\overrightarrow{Ac} \cdot \overrightarrow{AB}}{\| AB \|}.$$

The projection can be calculated as:

$$p = \begin{cases} A & \text{if } t < 0 \\ B & \text{if } t > 1 \\ A + t \cdot \overrightarrow{AB} & \text{if } 0 \leq t \leq 1 \end{cases}$$

For latent points near the tip principal points, we orthogonally project the latent point to the line segment formed by extending the tip principal point and its nearest neighbor principal point in the graph to obtain the projection point, i.e., $A + t \cdot \overrightarrow{AB}$.

We then calculate the distance between all of the projection points and construct a minimal spanning tree (MST) on the projection points. To avoid zero values of distance between cells projected to the same principal points, which prevents the calculation of a MST, the smallest positive distance between all cell pairs is added to all distance values. This MST is used to assign pseudotime for each cell (see this section below).

To encode the position of each cell within the branching structure of the trajectory, Monocle 2 performs a depth-first traversal of the principal tree learned during RGE. Without loss of generality, we assume that one principal point corresponds to one latent point. Following the definition introduced in Trapnell *et al.*[1], an ordering of cells (principal points) is obtained through a depth-first search (DFS) of the learned principal tree, starting from the root cell. We can then assign each cell to a trajectory segment, $b_x(G, \pi, i)$ which specifies the segment b_x by where the cell i is located based on the ordering list π and the graph structure G. We set $b_x = 1$ at the root cell and increase a segment counter b_x every time we reach a new branch point. More formally, we can write the formula of segment assignment as:

$$b_x(G,\pi,i) = \begin{cases} 1, & \text{if } i = 0 \\ \max(b_x(G,\pi,i)), j \preccurlyeq i, & \text{if} \left| E(G_j) \right| \leq 2 \\ \max(b_x(G,\pi,i)) + 1, j \preccurlyeq i, & \text{if} \left| E(G_j) \right| = 3 \end{cases}$$

where $j \preccurlyeq i$ represents all precedents j of i in the ordering π, $|E(G_i)|$ represents the degree of cell i. For the general cases in which the principal points are less than the cell numbers, cells inherit the segment assignment of their nearest principal point.

Similar to our previous definition of pseudotime[1], Monocle 2 calculates pseudotime based on the geodesic distance of each cell to the root cells on the MST of the projection points. For the pseudotime of cell i from a branching biological process s with branches given by b_x as $\phi_t(b_x, s_i)$, we can calculate its pseudotime recursively by adding the pseudotime of its parent cell on the MST of the projection points (closest cell on the same branch) with the Euclidean distance, $\| \vec{p}(b_x, s_i), \vec{p}(Parent(b_x, s_i)) \|_2$, between current cell and the parent on the MST, by setting the root cell as pseudotime 0. That is,

$$\phi_t(b_x, s_i) = \begin{cases} 0, & \text{if } b_x = 1,\ i = 0 \\ \phi_t(Parent(b_x, s_i)) + \| \vec{p}(b_x, s_i), \vec{p}(Parent(b_x, s_i)) \|_2, & \text{if } i > 0 \end{cases} \quad (5)$$

doi:10.1038/nmeth.4402

Assessing accuracy or robustness of pseudotime and branch assignments. We assessed the accuracy and robustness of each algorithm's pseudotime assignment against the reference ordering by two measures of correlation (Pearson's rho (default) and Kendall's tau) between their pseudotime values.

We used an adjusted Rand index (ARI)[23] value to measure the accuracy or robustness of tree segment assignment. Given the number of common cells, denoted as *S*, between the reference ordering and the ordering based on Monocle 2, Monocle 1, DPT, Wishbone, or SLICER (when available), and the corresponding trajectory segment assignments for reference ordering and ordering based on a different algorithm, $\mathcal{X}$ and $\mathcal{Y}$, namely, $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_r\}$ and $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_r\}$. The overlap between cells from segment i ($\mathcal{X}_i$) and cells from segment j ($\mathcal{Y}_i$) in each of the two orderings is represented by the number $n_{i,j}$ of cells in common, i.e., $n_{i,j} = |\mathcal{X}_i \cap \mathcal{Y}_i|$. We define the number of cells within segment i from reference ordering as $a_i = \sum_{j=1}^{s} n_{i,j}$, and the number of cells within segment j from ordering based on an algorithm is $b_j = \sum_{i=1}^{r} n_{i,j}$. The ARI value is then formulated as

$$ARI(\mathcal{X}, \mathcal{Y}) = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}$$

which is a measure of the similarity between trajectory segment assignments. When the ARI value is closer to 1, the segment assignment is more consistent between the two orderings.

To calculate the accuracy of pseudotime and branch assignment of the simulation (neuron/astrogenesis) and the Paul *et al.*[10] data set, the reference ordering corresponded to the real simulation, and branch assignments were based on manual assessment (see **Supplementary Note**) or the pseudotime and branch (or cell type suggested from the original study[18]) from the marker-based ordering (see next section).

We assessed the robustness of each algorithm via downsampling the simulation, the Paul *et al.*[10] blood data, and Treutlein *et al.*[8] lung data. We applied two different downsampling strategies. For the first strategy, we downsampled the full data set by selecting 80% of the cells 25 times without replacement. Then we ran Monocle 2, Monocle 1, DPT, and Wishbone to construct the branched trajectory. SLICER was excluded from the downsampling analysis because of its lengthy running times and instability on occasional downsample runs. Then we compared all pairs of downsamples by the metrics discussed above. For the second strategy, we progressively downsampled the full data set over a range of increasing fractions of cells from the full data set. Sampling was performed without replacement, and three different subsets were generated for each proportion to serve as replicates. Then we ran each algorithm, including Monocle 1, Monocle 2, DPT, and Wishbone, to construct branched trajectories for each fraction, which were compared to the corresponding trajectory that was built from the full data set. ARI, Pearson's rho, and Kendal's tau for all cases were then calculated as described above.

To assess the robustness of Monocle 2 over different parameter choices, we ran Monocle 2 and sampled a large range for each of the parameters used in DDRTree, including, *Dimension*, *lambda*, *maxIter*, *ncenter*, *param:gamm*, and *sigma*, while keeping other parameters as defaults, and we compared the result to the ordering obtained by running Monocle 2 with all default parameters. Pearson's rho and ARI were used to calculate the robustness.

Comparing different algorithms to a marker-based ordering. To test the accuracy of each trajectory-reconstruction algorithm, we compared the trajectories obtained to an empirical ordering based on marker genes. Relying on results from Paul *et al.*[10], we first selected *Pf4*, *Apoe*, *Flt3*, and *Cd74* as CMP-specific genes, *Hba-a2*, *Car2*, *Cited4*, and *Klf1* as MEP-specific genes, and *Mpo*, *Prg2*, *Prtn3*, and *Ctsg* as GMP-specific genes. Following the approach of Tirosh *et al.*[18], we then selected 100 other genes with expressions that correlated to these marker genes to calculate a stemness score and a GMP or megakaryocyte–erythroid progenitor (MEP) lineage score. We defined cells with a stemness score >0 as CMP cells, any cells with positive lineage were scores as MEP cells, and those with negative scores as GMP cells. This grouping of cells was used for branch assignment accuracy evaluation in **Supplementary Figure 9**. We then defined the reference pseudotime for each cell as:

$$\phi_t(b_x, s_i, l_i) = \begin{cases} d(C_i, 0), & \text{if } i \in \{CMP\} \\ Max_{j \in \{CMP\}}[d(C_i, 0) + d(C_i, 0)], & \text{otherwise} \end{cases}$$

where ϕ corresponds to the origin (0, 0), s_i corresponds to the stemness score, and l_i the lineage score for the lineage to which each cell is assigned, $d(C_i, 0)$ represents the Euclidean distance between cell $i(C_i)$ and the origin, and $\{CMP\}$ indicates the set of CMP cells.

Pseudotime correlations were computed on the paths from the root to each fate based on the reference ordering separately, and these were then averaged. Because the empirical ordering based on marker genes is not perfect, we also investigated the accuracy of the ordering in terms of the absolute lag-1 autocorrelation of fitted spline curve for the selected marker genes. We first selected the trajectory segments corresponding to the transition from the CMP cells to either the MEP or the GMP cells and then fit a kinetic curve for each marker gene for each transition with a spline curve with three degrees of freedom. We then calculated the absolute lag-1 autocorrelation *r*, which is defined as following:

$$r = \frac{\sum_{i=1}^{N-1} |(Y_i - \mu)(Y_{i+1} - \mu)|}{\sum_{i=1}^{N} (Y_i - \mu)^2}$$

where ϕ_i represents the gene expression at time stamp *i*, and μ is the mean expression across the pseudotime series for that gene. Higher autocorrelation values implied smoother gene expression dynamics based on the ordering.

Although a reference ordering based on markers from the literature can serve as a reasonable gold standard, it also introduces bias in a benchmarking analysis. Algorithms that order cells based on a small set of informative genes (which include or correlate with the marker genes) will likely match it better than algorithms that order cells based on all genes. We therefore explored orthogonal means of measuring accuracy of each program's ordering based on the neuron simulation data (**Supplementary Note**).

Reconstructing the complex hematopoiesis hierarchy. We used a scree plot to select ten dimensions as the intrinsic dimensionality of the developmental trajectory for the Paul *et al.*[10] data set (cells used in Fig. 1 of the original study[10]). Five branch points and six terminal lineages (monocytes, neutrophils or eosinophil, basophils, dendritic cells, megakaryocytes, and erythrocytes) were revealed. We ordered the cells using the genes Paul *et al.* used to cluster their data rather than the genes from dpFeature, for the sake of consistency with their clustering analysis. Similarly, we reconstructed the Olsson *et al.*[9] data sets in four dimensions. The major bifurcation between the granulocyte and monocyte branch (GMP), as well as the intricate branch between the GMPs and the megakaryocytes and erythrocyte (Ery/Meg) were revealed. The top 1,000 genes from dpFeature, based on the WT cells, were used in the entire data set. The distribution of cells across clusters from the original paper over each segment of the principal graph were calculated and visualized in the heat map.

We applied BEAM analysis to identify genes that significantly bifurcated between the Ery/Meg and GMP branch on the Olsson *et al.*[9] data set from the WT cells. We then calculated the instantaneous log ratios (ILRs) of gene expression between the Ery/Meg and GMP branch and found genes with a mean ILR >0.5. The ILRs were defined as:

$$ILR_t = \log\left(\frac{Y_1^t}{Y_2^t}\right)$$

Thus, ILR_t is calculated as the log ratio of the fitted value at the interpolated pseudotime point t for the Ery/Meg lineage and that for the GMP lineage. The average expression values of the genes that were found to bifurcate were used to calculate the lineage score for both the Olsson *et al.* and Paul *et al.* data sets (**Supplementary Note**). The same genes were used to create the multi-way heat map for both the Paul *et al.* and Olsson *et al.* data sets.

In addition, pseudotime-dependent genes for the Ery/Meg and GMP branch were identified in the WT data set from the Olsson *et al.* study. Genes expression levels that were higher in progenitor cells than in more differentiated cells were selected as 'stemness' markers and averaged to produce a stemness score for both the Olsson *et al.* and the Paul *et al.* data sets.

Code availability. The version of Monocle 2 (version: 2.2.0) used in this study is provided as **Supplementary Software**. The newest Monocle 2 is available through Bioconductor, as well as GitHub (https://github.com/cole-trapnell-lab/monocle-release). DDRTree, SimplePPT, and SGL-tree/L1 graph were implemented in DDRTree (version: 0.1.5), SimplePPT (version 0.1.0) and L1Graph (version: 0.1.0), respectively (available (DDRTree) or will be (simplePPT, L1Graph) available from CRAN). The density-peak algorithm is available from https://github.com/Xiaojieqiu/densityClust/tree/knn_dp (densityClust: version 0.3). All those packages are included in **Supplementary Software**, which also includes a helper package, xacHelper, which contains helper functions, as well as the other analysis code that can be used to reproduce all of the figures and data in this study. Jupyter notebooks, which were used to reproduce the analysis-related data sets from the Olsson *et al.*[9] and Paul *et al.*[10] studies, are included in **Supplementary Software** as well. In addition, we deposited the same data at https://github.com/cole-trapnell-lab/monocle2-rge-paper.

Data availability. Four public scRNA-seq data sets were used in this study: the HSMM data set, GSE52529 (ref. 1); the lung data set, GSE52583 (ref. 8); the Paul *et al.* data set[10], http://compgenomics.weizmann.ac.il/tanay/?page id=649; and the Olsson data set[9], synapse ID syn4975060. Data for neuron simulation, results of the least-action paths, as well as the complicated tree structure, are included as **Supplementary Data 1–3**, respectively. A **Life Sciences Reporting Summary** for this paper is available.

21. Mao, Q., Yang, L., Wang, L., Goodison, S. & Sun, Y. . in *Proceedings of the 2015 SIAM International Conference on Data Mining* 792–800 (Society for Industrial and Applied Mathematics, 2015).
22. Boyd, S. & Vandenberghe, L. *Convex Optimization* (Cambridge University Press, Cambridge, 2004).
23. Rand, W.M. *J. Am. Stat. Assoc.* **66**, 846–850 (1971).

doi:10.1038/nmeth.4402

nature research

Corresponding author(s): Cole Trapnell

☐ Initial submission ☒ Revised version ☐ Final submission

Life Sciences Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form is intended for publication with all accepted life science papers and provides structure for consistency and transparency in reporting. Every life science submission will use this form; some list items might not apply to an individual manuscript, but all fields must be completed for clarity.

For further information on the points included in this form, see Reporting Life Sciences Research. For further information on Nature Research policies, including our data availability policy, see Authors & Referees and the Editorial Policy Checklist.

Experimental design

1. Sample size

Describe how sample size was determined.

We analyzed public datasets in this manuscript.

2. Data exclusions

Describe any data exclusions.

Low quality cells from the HSMM, Lung, Olsson and Paul datasets were excluded for downstream analysis. Full details on the criteria for removing those cells are included in "Details on analyzing datasets used in this study" of the Supplementary Notes. Exact scripts used to perform those filtering are included in the Supplementary Software.

3. Replication

Describe whether the experimental findings were reliably reproduced.

We analyzed public datasets in this manuscript

4. Randomization

Describe how samples/organisms/participants were allocated into experimental groups.

We analyzed public datasets in this manuscript

5. Blinding

Describe whether the investigators were blinded to group allocation during data collection and/or analysis.

We analyzed public datasets in this manuscript

Note: all studies involving animals and/or human research participants must disclose whether blinding and randomization were used.

6. Statistical parameters

For all figures and tables that use statistical methods, confirm that the following items are present in relevant figure legends (or in the Methods section if additional space is needed).

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement (animals, litters, cultures, etc.)
☐	☒ A description of how samples were collected, noting whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ A statement indicating how many times each experiment was replicated
☐	☒ The statistical test(s) used and whether they are one- or two-sided (note: only common tests should be described solely by name; more complex techniques should be described in the Methods section)
☐	☒ A description of any assumptions or corrections, such as an adjustment for multiple comparisons
☐	☒ The test results (e.g. *P* values) given as exact values whenever possible and with confidence intervals noted
☐	☒ A clear description of statistics including central tendency (e.g. median, mean) and variation (e.g. standard deviation, interquartile range)
☐	☒ Clearly defined error bars

See the web collection on statistics for biologists for further resources and guidance.

Nature Methods: doi:10.1038/nmeth.4402

Software

Policy information about availability of computer code

7. Software

Describe the software used to analyze the data in this study.	A tarball comprises of a version of Monocle 2 (version: 2.2.0), DDRTree (version: 0.1.5), simplePPT (version 0.1.0) and L1Graph (version: 0.1.0), densityClust (version 0.3), a helper package (xacHelper) containing helper functions as well as all analysis code which can be used to reproduce all figures and data in this study are included along with this manuscript. All packages or analysis scripts are written by the authors of this work.

For manuscripts utilizing custom algorithms or software that are central to the paper but not yet described in the published literature, software must be made available to editors and reviewers upon request. We strongly encourage code deposition in a community repository (e.g. GitHub). *Nature Methods* guidance for providing algorithms and software for publication provides further information on this topic.

Materials and reagents

Policy information about availability of materials

8. Materials availability

Indicate whether there are restrictions on availability of unique materials or if these materials are only available for distribution by a for-profit company.	All data used in this study is available on a github repo (https://github.com/cole-trapnell-lab/monocle2-rge-paper) for this work as well as included in the supplementary data. The github repo also includes jupyter notebooks to reproduce analysis performed for the Paul and Olsson dataset in this work.

9. Antibodies

Describe the antibodies used and how they were validated for use in the system under study (i.e. assay and species).	No antibodies are used for this work

10. Eukaryotic cell lines

a. State the source of each eukaryotic cell line used.	No eukaryotic cell lines are used for this work
b. Describe the method of cell line authentication used.	Not applicable
c. Report whether the cell lines were tested for mycoplasma contamination.	Not applicable
d. If any of the cell lines used are listed in the database of commonly misidentified cell lines maintained by ICLAC, provide a scientific rationale for their use.	Not applicable

Animals and human research participants

Policy information about studies involving animals; when reporting animal research, follow the ARRIVE guidelines

11. Description of research animals

Provide details on animals and/or animal-derived materials used in the study.	Not applicable

Policy information about studies involving human research participants

12. Description of human research participants

Describe the covariate-relevant population characteristics of the human research participants.	Not applicable

Nature Methods: doi:10.1038/nmeth.4402