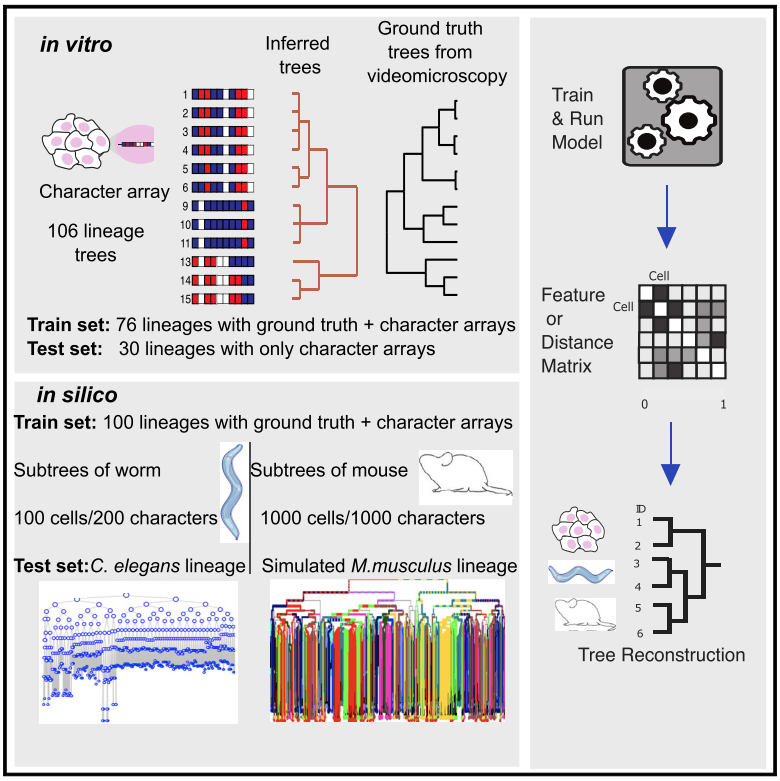
Cell Systems

Article

Benchmarked approaches for reconstruction of *in vitro* cell lineages and *in silico* models of *C. elegans* and *M. musculus* developmental trees

Graphical abstract



Highlights

- We organized a DREAM challenge to benchmark methods of cell lineage reconstruction
- Using experimental, *in silico* datasets as ground-truth trees of 10^2, 10^3, and 10^4 cells
- Smaller trees allowed the training of a machine-learning decision tree approach
- These results delineate a potential way forward for solving larger cell lineage trees

Authors

Wuming Gong, Alejandro A. Granados, Jingyuan Hu, ..., Ehud Shapiro, Michael B. Elowitz, Pablo Meyer

Correspondence

pmeyerr@us.ibm.com

In brief

Organisms can be composed of trillions of cells that derive from a single one through repeated rounds of cell divisions. Knowing the lineage relationships between the cells of a fully developed organism—its cell lineage—provides the framework for understanding when, where, and how cell fate decisions are made. The reconstruction of cell lineage trees from molecular data share similarities with evolutionary trees, but having the solution of the tree allowed us for the first time to compare the performance of a new set of methods for reconstructing cell lineage trees.

Gong et al., 2021, Cell Systems *12*, 810–826
August 18, 2021
https://doi.org/10.1016/j.cels.2021.05.008

CellPress

CellPress
OPEN ACCESS

Cell Systems

Article

Benchmarked approaches for reconstruction of *in vitro* cell lineages and *in silico* models of *C. elegans* and *M. musculus* developmental trees

Wuming Gong,[1,22] Alejandro A. Granados,[2,22] Jingyuan Hu,[3,22] Matthew G. Jones,[4,5,22] Ofir Raz,[6,22] Irepan Salvador-Martínez,[7,22] Hanrui Zhang,[8,22] Ke-Huan K. Chow,[2] Il-Youp Kwak,[9] Renata Retkute,[10] Alidivinas Prusokas,[11] Augustinas Prusokas,[12] Alex Khodaverdian,[4] Richard Zhang,[4] Suhas Rao,[4] Robert Wang,[4] Phil Rennert,[13] Vangala G. Saipradeep,[14] Naveen Sivadasan,[14] Aditya Rao,[14] Thomas Joseph,[14] Rajgopal Srinivasan,[14] Jiajie Peng,[15] Lu Han,[15] Xuequn Shang,[15] Daniel J. Garry,[1] Thomas Yu,[16] Verena Chung,[16] Michael Mason,[16] Zhandong Liu,[3] Yuanfang Guan,[8] Nir Yosef,[4] Jay Shendure,[17,18,19,20] Maximilian J. Telford,[7] Ehud Shapiro,[6] Michael B. Elowitz,[2] and Pablo Meyer[21,23,*]

[1]Lillehei Heart Institute, University of Minnesota, 2231 6th St S.E, 4-165 CCRB, Minneapolis, MN 55114, USA
[2]California Institute of Technology, Pasadena, CA 91125, USA
[3]Program in Quantitative and Computational Biosciences, Baylor College of Medicine, Houston, TX 77030, USA
[4]Department of Electrical Engineering & Computer Science, University of California, Berkeley, Berkeley, CA, USA
[5]Integrative Program of Quantitative Biology, University of California, San Francisco, San Francisco, CA, USA
[6]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 761001, Israel
[7]Centre for Life's Origins and Evolution, Department of Genetics, Evolution and Environment, University College London, Gower Street, London WC1E 6BT, UK
[8]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI 48109, USA
[9]Department of Applied Statistics, College of Business & Economics, Chung-Ang University, 84, Heukseok-ro, Dongjak-gu, Seoul, Republic of Korea
[10]Department of Plant Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EA, UK
[11]School of Natural and Environmental Sciences, Newcastle University, Newcastle NE1 7RU, UK
[12]Department of Life Sciences, Imperial College London, London SW7 2AZ, UK
[13]EC Wise Inc., 1299 4th St #505, San Rafael, CA 94901, USA
[14]TCS Research and Innovation, Tata Consultancy Services, Hyderabad 500019, India
[15]School of Computer Science, Northwestern Polytechnical University, Xi'an, China
[16]Sage Bionetworks, 2901 3rd Ave #330, Seattle, WA 98121, USA
[17]Department of Genome Sciences, University of Washington, Seattle, WA 98195, USA
[18]Allen Discovery Center for Cell Lineage Tracing, Seattle, WA, USA
[19]Brotman Baty Institute for Precision Medicine, Seattle, WA, USA
[20]Howard Hughes Medical Institute, Seattle, WA, USA
[21]T.J. Watson Research Center, IBM, Healthcare & Life Sciences, 1101 Kitchawan Rd 10598, Yorktown Heights, NY 10598, USA
[22]These authors contributed equally
[23]Lead contact
*Correspondence: pmeyerr@us.ibm.com
https://doi.org/10.1016/j.cels.2021.05.008

SUMMARY

The recent advent of CRISPR and other molecular tools enabled the reconstruction of cell lineages based on induced DNA mutations and promises to solve the ones of more complex organisms. To date, no lineage reconstruction algorithms have been rigorously examined for their performance and robustness across dataset types and number of cells. To benchmark such methods, we decided to organize a DREAM challenge using *in vitro* experimental intMEMOIR recordings and *in silico* data for a *C. elegans* lineage tree of about 1,000 cells and a *Mus musculus* tree of 10,000 cells. Some of the 22 approaches submitted had excellent performance, but structural features of the trees prevented optimal reconstructions. Using smaller sub-trees as training sets proved to be a good approach for tuning algorithms to reconstruct larger trees. The simulation and reconstruction methods here generated delineate a potential way forward for solving larger cell lineage trees such as in mouse.

INTRODUCTION

Lineage inference for understanding development

A fundamental challenge in biology is the reconstruction of the developmental histories of cells as they divide and progress through differentiation into different cell types. Indeed, multicellular organisms can be composed of billions or trillions of cells that derive from a single cell through repeated rounds of cell division. Knowing the lineage relationships between the cells of a fully developed organism—its cell lineage—would provide a

framework to understand when, where, and how cell fate decisions are made. Further, it can also be useful to understand the progression of disease such as in tumor subclonal reconstruction (Salcedo et al., 2020) or the development of an organ such as the brain (Evrony et al., 2015; Lodato et al., 2015). Historically, lineages of individual cells have only been fully reconstructed by their direct observation through microscopy as for the nematode *Caenorhabditis elegans* (Sulston and Horvitz, 1977). This direct observation approach is however not possible for most animals as the cells are not visible (Livet et al., 2007). In the 1980s new methods allowed marking all the descendants of a single cell by the injection of a dye or the expression of a marker gene. Since then, many new methods have been devised to improve cell lineage tracking, including inducible recombinases (Kretzschmar and Watt, 2012), fluorescent or genetic reporters (Kebschull and Zador, 2018; Weissman and Pan, 2015), or a combination of both (Garcia-Marques et al., 2020). However, these approaches come at the cost of resolution, meaning that lineage relationships of individual cells are not fully recovered.

Recent advances in sequencing technologies have enabled a variety of RNA-based methods to infer differentiation trajectories in multiple organisms and cell types by ordering the changes in single-cell gene expression along a pseudo-time axis representing the progression through differentiation (Wagner and Klein, 2020). However, these methods focus on the expression profiles of cells but do not have access to their genealogical relationships. In this regard, somatic mutations accumulated during normal development have been used to reconstruct genetic lineages (Behjati et al., 2014; Frumkin et al., 2005) and for example trace mosaicism in the brain (Evrony et al., 2015; Lodato et al., 2015). Deep sequencing of cDNA from T cell receptors has also been used to establish clonal development of T cells (Becattini et al., 2015). Cell lineage inference has also been done using copy-number variations, structural markers such as SNVs, indels, retrotransposon elements, microsatellite repeats, as well as epigenetic markers such as DNA methylation (Kester and Oudenaarden, 2018).

New lineage recording technologies

Recently, the advent of CRISPR-based molecular tools have produced a new generation of lineage reconstruction approaches inspired by principles of phylogenetic inference using naturally occurring DNA mutations. The DNA-editing technologies have been applied to introduce mutations in the genetic material of cells such that a registry of their genetic relationships is recorded and available for readout by sequencing (Alemany et al., 2018; Chan et al., 2019; McKenna et al., 2016; Perli et al., 2016; Spanjaard et al., 2018). Indeed, the inserted synthetic construct can accumulate stochastic mutations upon induction of CRISPR-Cas9 activity as cells differentiate during development with the goal of resolving cellular lineages of complex model organisms (McKenna et al., 2016; Wagner and Klein, 2020). Different versions of CRISPR-based methods such as scGESTALT, LINNAEUS and ScarTrace techniques have been successfully used to investigate cellular lineages in various animal models (Alemany et al., 2018; McKenna and Gagnon, 2019; Raj et al., 2018; Spanjaard et al., 2017). At the same time, other types of lineage recording techniques have been applied to allow readout by *in situ* imaging that enables lineage analysis through the maintenance of the spatial information (Chow et al., 2021; McKenna and Gagnon, 2019). Some of these approaches have applied phylogenetic reconstruction algorithms to infer the cell lineage, while others developed ad hoc cell lineage reconstruction algorithms, but this explosion of lineage tracing technologies has increased the urgency for new reconstruction methods (Salvador-Martínez et al., 2019).

In principle, as in phylogenetic tree reconstruction (Frieda et al., 2017; McKenna et al., 2016), the recorded mutations should encode enough information enabling inference of the likely tree structures that could represent the actual lineage relationships. However, there are significant challenges for tree inference when applying standard phylogenetic methods to lineage recordings. The main limitations include noise from the experimental readout, restrictions in the total available "DNA memory" for recording, and the random convergence of identical edit patterns in non-related cells, or homoplasy (Salvador-Martínez et al., 2019). It also remains unclear whether machine-learning algorithms that go beyond classical phylogenetic methods, such as neighbor joining or maximum parsimony, could consistently reconstruct cell lineages with higher accuracy. While phylogenetic methods typically analyze a relatively small number of species and many more DNA sites, genes, or even whole genomes (McKenna and Gagnon, 2019), CRISPR-based lineage recording aims to capture hundreds to thousands of cells with the compromise of limited numbers of editable sites. Additional limitations include variability in mutation rates for each site, large nucleotide deletions resulting in sequence dropouts, and single deletions that can erase previous mutations or ablate multiple targets. Although maximum parsimony-based methods have shown initial success when applied to lineage tracing (McKenna and Gagnon, 2019; McKenna et al., 2016; Price et al., 2010), the key differences discussed above make it challenging to directly apply phylogenetic methods to lineage tracing data.

After having performed lineage tree inference one would ideally like to evaluate the reconstruction accuracy, however for most of these technologies the ground truth is inaccessible, meaning that we do not know the actual lineage relationships. Indeed, with rare exceptions (Sugino et al., 2019), to date no lineage reconstruction approach has been rigorously examined for its performance/robustness across diverse molecular tools, DNA-based recording methods, datasets, number of cells, topology of lineage trees and diverse metrics used for evaluation. Given the lack of benchmarking, there is still no agreement regarding the best practices for inferring cellular lineages from the recording datasets generated with these recently developed molecular tools.

The DREAM initiative

To catalyze the development of new methods to perform lineage reconstruction, we organized the Allen institute lineage reconstruction DREAM challenge, which ran from October 2019 through February 2020. DREAM (Dialogue for Reverse Engineering and Methods) challenges are a platform for crowdsourcing collaborative competitions where a rigorous evaluation of each submitted solution allows for objective comparison and assessment of their performance (Saez-Rodriguez et al., 2016). The value of DREAM resides not only in the acceleration of research

through the participation of many teams while solving a common problem, but just as importantly, in the diversity of approaches used and the quality and reproducibility of each provided solution to problems in emerging areas of biology. The aggregation of the individual solutions, i.e., the different approaches and insights to a common problem, namely the "wisdom of the crowds," leads to a generally superior performance than any individual solution, from where collective insights can be garnered.

The DREAM challenge for lineage reconstruction

The lineage reconstruction DREAM challenge aimed to provide a new perspective on lineage inference by enabling participants from diverse fields to submit their reconstruction of trees for which the ground truth, i.e., the actual lineage, existed but was not provided. It consisted of three challenges with lineages of increasing numbers of cells. The first challenge leveraged a then unpublished experimental dataset of 106 trees recorded with intMEMOIR in mouse embryonic stem cell colonies of less than 100 cells (Chow et al., 2021). This technique was chosen as it has the key advantage of readout by imaging that can be coupled with a time-lapse movie of the cells as they divide to provide a ground-truth lineage tree (Figure 1A). In the second challenge participants had to reconstruct an *in silico* tree of 1,000 cells, whose topology was derived from the *Caenorhabditis elegans* developmental cell lineage tree by removing a few clades in order to mask its identity to the participants. A general framework for simulation of CRISPR-based lineage recording (Figure 1B) (Salvador-Martínez et al., 2019) was used to simulate mutations in a recording array on top of the resulting topology (see Figure 1C). In the third challenge, participants had to infer the lineage of cells in a simulated tree of ~10,000 cells (Figure 1D) representing 11 different cell types after 1 year of *M. musculus* development (Figure 1E). Simulating such a large tree was made possible by applying the environment-dependent stochastic tree grammars (eSTGt), a programing and simulation environment for population dynamics (Spiro and Shapiro, 2016) adapted to simulate cell lineages (see STAR Methods). While the size of the actual simulated tree is estimated to be about 10^{12} or a trillion cells, the final sub-sampled lineage stored information for only 10,000 cells (see Figure S1).

Experimental *in vitro* dataset

intMEMOIR is a synthetic image-readable lineage recording system that has been recently developed and tested in mouse embryonic stem cells and the brain of *Drosophila melanogaster* (Chow et al., 2021). This technology builds upon a previously developed recording system named MEMOIR (memory by engineered mutagenesis with optical *in situ* readout) (Frieda et al., 2017). In its current implementation, intMEMOIR consists of a multi-state memory DNA array that can be edited irreversibly by serine integrases and integrated at defined genomic sites. While MEMOIR's design enabled 2 different states for each recording unit in the memory array, intMEMOIR enables 3 different states. Upon induction by doxycycline, the serine integrase Bxb1 can bind to the editable character array elements or bar codes, and by DNA-recombination mutate the recording element ground state (represented as "1") into either two possible states, a deletion (represented as "0") or an inversion (represented as "2") of the DNA sequence. The recording process is fully stochastic and happens irreversibly at a constant rate, as any element in the array can be edited at any moment. On mouse embryonic stem cells, Chow et al. showed that lineage information can be recorded irreversibly and stored in the intMEMOIR array, while also readout using microscopy. From the recorded data, the lineage history can then be inferred (Figure 1A).

In the experiment, the growth of 106 cell colonies was traced, each one started from an individual cell carrying an unedited 10-character array. Recording was induced for the first 36 h of growth (approximately 3 cell divisions), and cells were then allowed to grow with no further recording for an additional 24 h. At this point the arrays for each cell in the colony were readout using single molecule fluorescent *in situ* hybridization (smFISH). For each colony, the ground-truth lineage was obtained from time-lapse movies. As cells grow at different speeds and some of them die, the resulting colonies had a distribution of sizes, from 4 to 39 cells (see Table S1).

Simulated *in silico* datasets

To complement the challenge datasets, data from simulated recording arrays, with respectively 200 Cas9 targets in each cell for *C. elegans* and 1,000 targets for *M. musculus*, were generated. Inspired by the GESTALT technique (McKenna et al., 2016), in the simulations, every cell is represented as a vector of 200 (or 1,000) characters, each character representing one Cas9 target. The simulations started with one cell, the fertilized egg, and all its targets in an unmutated ground state represented with "0" (see Figure 1C) had the possibility to change to either of 30 different mutational outcomes stochastically as cells divide (see Box 1). The initial cell then undergoes a series of cell divisions growing into a population of ~1,000 cells for *C. elegans* and about a trillion cells from which ~10,000 cells are preserved for *M. musculus* (see STAR Methods). The recording array accumulates independent and irreversible CRISPR-induced mutations with a constant probability per time unit, inherited in subsequent cell divisions (see Box 1).

When a Cas9-induced mutation occurs, the double strand of DNA is broken, which is eventually repaired by the cell. However, in cases where two or more relatively close double strands break before the cell repair machinery can act, the DNA between these breaks can be lost and such events are called an "inter-target deletions." To make these simulations more realistic, we included inter-target deletions affecting 5%–10% of the mutation events (see STAR Methods and Box 1). We also introduced different probabilities for the different mutational outcomes, in agreement with experimental evidence (McKenna and Gagnon, 2019). Additionally, for the *M. musculus* simulations we implemented a 20% data acquisition dropout to reflect the fact that the data acquisition from single cells is rarely perfect (Qiu, 2020) (see Box 1). In summary, we introduced experimental parameters where possible in the simulation in order to approximate realistic recording assays.

Training data

As the goal of these challenges was not only to benchmark cell lineage reconstruction algorithms but also to mobilize a larger community for evaluating new optimal tree-building methods, we provided training data for each challenge. In the *in vitro*

CellPress
OPEN ACCESS

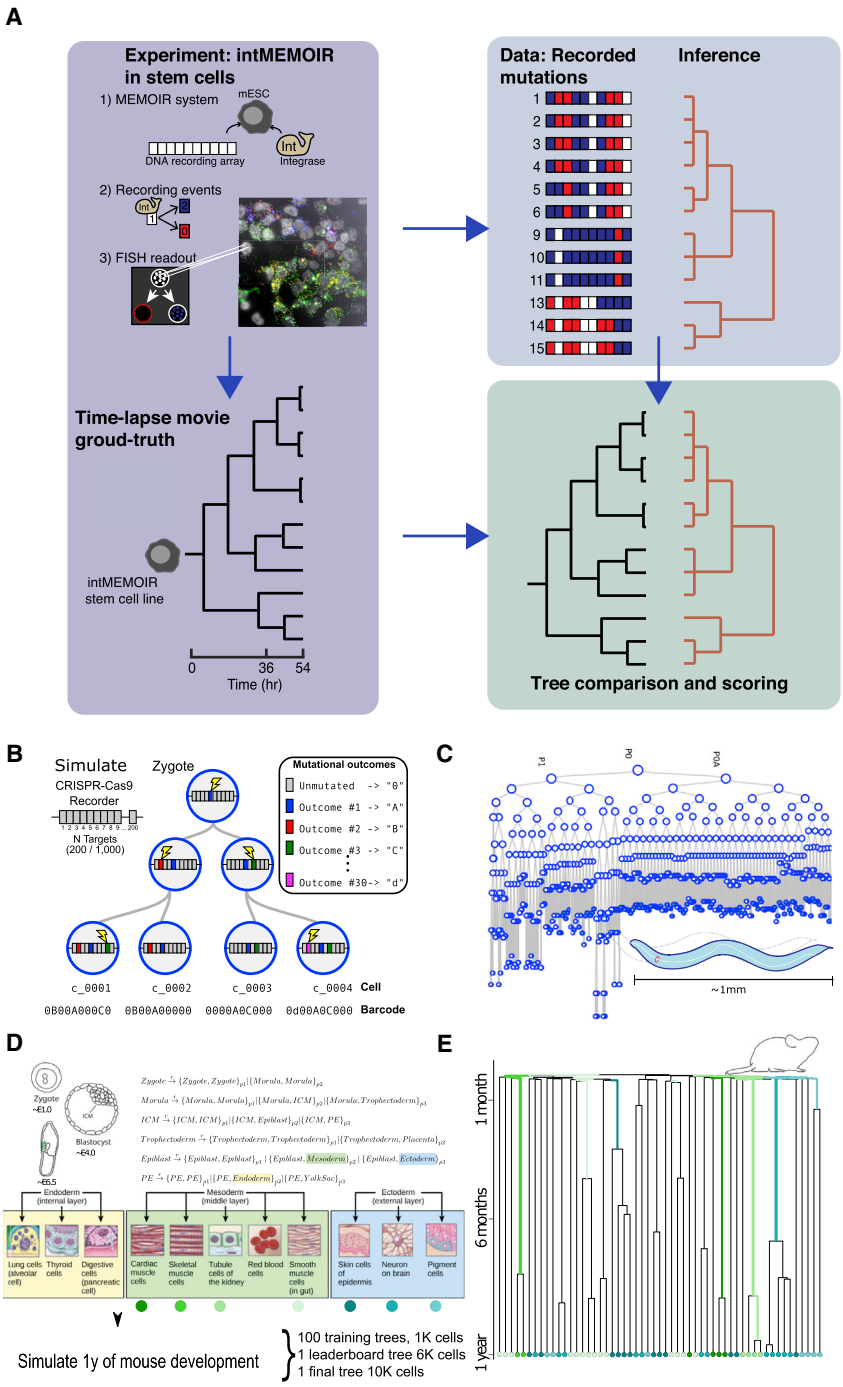


Figure 1. Three challenges for lineage reconstruction from experimental and *in silico* generated character arrays

(A) Challenge consisting of reconstructing *in vitro* growing cell lineages. The lineage tracing intMEMOIR system consists of a character array of editable DNA elements—or bar codes—and the integrase enzyme Bxb1. A mouse stem cell line was engineered with both components. A recording event happens when the integrase stochastically edits one of the 10 elements in the array, resulting in two possible outcomes, deletion and inversion (blue and red squares). As cells divide, each individual daughter cell acquires unique edit patterns (right panel). Finally, *in situ* readouts by smFISH enables the extraction of recorded data for individual cells. Since the whole experiment is done under a microscope, a ground-truth lineage tree is also generated which we use as our ground truth.

(B) Diagram showing the simulations performed to generate the character arrays for the two *in silico* datasets, an initial cell with N multiple targets (200 or 1,000 for the *C. elegans* or *M. musculus* challenge, respectively) accumulates one of the 30 independent mutations with a given probability, which are inherited in subsequent cell divisions. The pattern of mutations accumulated in each cell is used to infer the lineage tree.

(C) *In silico* challenge consisting of reconstructing the ~1,000 cells *C. elegans* cell lineage from the simulated cell character arrays. For visualization purposes the ground-truth cell lineage shows only the first 9 cell divisions.

(D) Challenge consisting of reconstructing ~10,000 cells from a simulated *M. musculus* cell lineage developmental tree generated using stochastic tree grammar (STG). The tree simulation describes the early stages of mouse development up to the three germ layers (mesoderm, ectoderm, and endoderm are highlighted with colors in the equations and resulting tree), those in turn continue to differentiate to the final populations of about 1,012 cells and 11 cell types simulated in the challenge.

(E) Displayed is a simulation example of the ground-truth tree for a subset of cells from the mesoderm and ectoderm, highlighted with the respective colors, throughout 1 year of development. The edges width and color reflect the hypergeometric score of its descending leaves.

challenge, participants were asked to reconstruct the test dataset consisting of 30 cell colonies using only the intMEMOIR array readout, as the ground truth for these lineages was not accessible to the participants. As training set, participants were given array readout data from 76 colonies along with the corresponding ground-truth lineages (Box 1).

For the *in silico* challenges, the training data included the ground-truth simulations of 100 lineage trees and their mutated array states. These trees comprised 100 cells for *C. elegans* and 1,000 cells for *M. musculus* generated with the same simulation scheme as for the whole *C. elegans* and *M. musculus* trees. The rationale was to test whether training sets composed of smaller trees could still be helpful to fine-tune algorithms then used to reconstruct larger lineages. The *C. elegans* training set tree topology was generated by 100 iterations of pruning and regrafting sub-trees of 100 cells from the whole animal lineage tree (Box 1), to preserve some of the initial topology without giving away the origin of the tree. We indeed verified that the aggregation of the 100 trees given for training showed no direct similarity to the 1,000 cells *C. elegans* tree. The *M. musculus* training set was obtained

Box 1. Training set

One of the main goals of this challenge was to provide participants with a training set composed of several trees, their cell's character arrays and the gold standard tree solution. This allowed participants to train or optimize their methods.(A) In the *in vitro* experiments to obtain mouse stem cell lineages, mutations were induced for the first 36 h of growth (approximately 3 cell divisions), and cells were then allowed to grow with no further changes in the recording arrays for an additional 24 h. For all these cells the final values (unmodified encoded as 1, inverted encoded as 2, or deleted encoded as 0) of the 10 character arrays were obtained by smFISH, while cell divisions were tracked by video microscopy (see Table S1). Two partitions were created from the original unpublished dataset containing 106 lineages, which represent sufficient experimental data to extract a training set: the training partition composed of 76 trees was provided for the teams along with the corresponding ground-truth lineages, for the test partition composed of 30 trees only the cells character arrays were provided without ground truth. The partitions were defined to have similar tree size distribution, given that the lineages were composed of a different number of cells depending on the cell division and survival rates, shown in middle histogram panel. Also, a similar median RF score distribution between the two datasets when using a maximum-likelihood method described in Chow et al. was used as partition criteria, see box plot on bottom panel. (B) For the *in silico* challenges, both character arrays for the training and test sets were simulated in a similar way. The type of Cas9-induced mutations consisted of 32 characters "A" to "Z" and "a" to "e" and character deletion "-." The characters represent DNA targets for Cas9, but no specific relationship with actual DNA sequences was established. The starting character was "0" and the probability of mutating to one of the 30 characters or of being deleted (insertions were not considered) followed in alphabetical order the Gamma probability distribution used to sample the mutations, shown in blue, and in red a fit on the histogram of the actual results. Mutations are irreversible, once a target is mutated, it can no longer change, either to revert to the unmutated state or to transit to a new state.(C) Inter-target deletions were simulated for both *in silico* challenges where *C. elegans* arrays were composed of 100 characters and *M. musculus* of 1,000 characters. When a Cas9-induced mutation occurs, the double strand of DNA is broken, which is eventually repaired by the cell. However, in cases where 2 or more relatively close double strands break before the cell repair machinery can act, the DNA between these breaks can be lost this is known as an "inter-target deletion." We implemented these so that when two mutations occur in close targets (less than 20 targets apart in the recording array) within a short interval of time during a given cell division, all the targets between them are removed. In these simulations, 5%–10% of targets are missing due to inter-target deletions. (D) Acquisition dropout distributions were implemented only for the *M. musculus* challenge. In order to capture the variability of the signal quality in both the individual samples and the different sites we modeled the "sequencing dropout" of single-cell samples by assigning distinct coverage factors for each sample and for each locus. The density of cell coverage factors P = (*pi*: I = 1 to M) is the probability of obtaining a signal in each sample or and the density of site coverage factors Q = (*qj*: j = 1 to N) as the probability of obtaining a signal in each locus. The probability of obtaining a signal in sample I and locus j thus equals *pi.qj.r*. Those are multiplied to get the individual coverage factor of a specific site in a specific cell, finally deriving the acquisition dropout status as a factor of a global coverage parameter *r*. (E) We provided 100 training cell lineage trees of 100 cells for *C. elegans* and of ~1,000 cells for *M. musculus*. As the *C. elegans* tree has been experimentally solved, its topology was used to generate the training set. The *M. musculus* tree being completely synthetically generated, the training set was obtained by simply running shorter simulations to obtain ~1,000-cells trees instead of the ~10,000-cells tree for the test set.(F) Top: we extracted the *C. elegans* training set from its tree topology by cutting and pasting subsets of tree branches. We followed the indicated schematic of cutting and pruning hundred times subsets of the whole tree. Note that only one prune and regraft event is shown in red in the diagram. From the obtained topology, the mutation arrays were generated from the Gamma distribution and then 100 cells were sampled. This process was repeated 100 times to obtain a full training set. Bottom: the boxplots show the performance of each submitted method for inferring the lineage trees from 100 training lineages used in the *C. elegans in vitro* challenge. The similarity between the inferred trees and the ground-truth trees was measured by Robinson-Foulds distance left and Triplet distance right. Red stars indicate the score for the *C. elegans* 1,000 cell tree. The values for the *M. musculus* training set were not established due to excessive computational time required.

(Continued on next page)

using the same eSTG algorithm used for the test dataset but ran for a shorter time in order to obtain smaller trees of 1,000 cells. Importantly, the *M. musculus* challenge also had an intermediate step where participants could submit solutions to a ~6,000 cell tree and obtain their scoring results on a leaderboard in real time. The leaderboard encouraged participation through competition and provided a way of testing the scalability of the approaches. For scoring, the submitted lineage tree inferences for the test dataset were then compared with their corresponding ground truth using two different metrics (see Box 2).

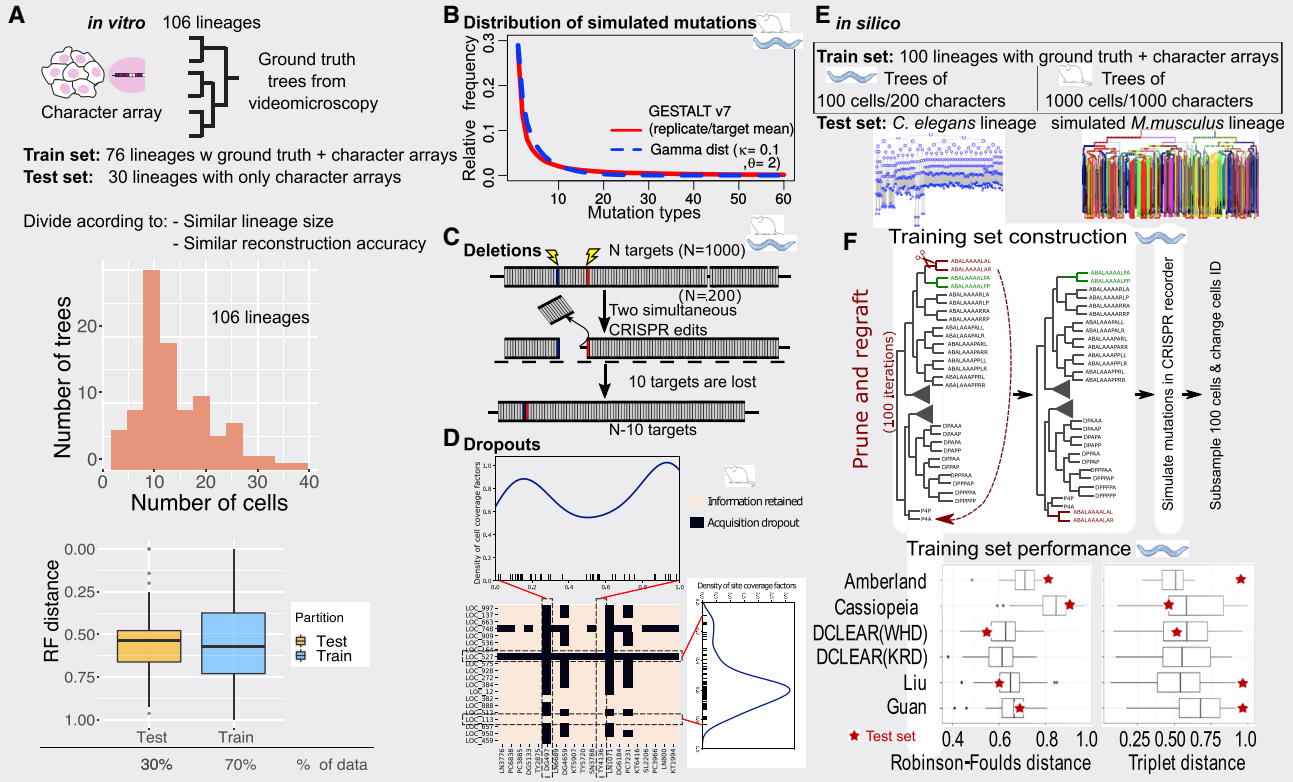
RESULTS

Best performing methods

Overall, the challenge was successful in its main goal to attract a variety of approaches and teams, as twenty-two submissions were received in total for the three challenges. Figures 2A–2C show the score rankings by both the Robinson-Foulds (RF) and triplet distances. For the *in vitro* challenge, where nine teams participated, it is clear that the diverse set of approaches reached a plateau in performance for both metrics, which suggests that participants successfully extracted and used all

CellPress
OPEN ACCESS

Box 1. *Continued*



available information in the data (Figures 2A, S2, S3A, and S3B fitted blue line to the medians). We found that the top three teams performed equally well even when calculating the Bayes factor and an additional quartet metric (Figure S2). Interestingly, the two distance metrics generated different rankings, showing that while correlated the two metrics are not identical. We noted that in general teams performed better on the RF distance compared with the triplet metric (Figures 2A and S3C). This indicates that for trees less than 100 cells, the triplet metric is more stringent than the whole-tree partitions measured by RF.

Five teams submitted solutions for the *C. elegans* and three teams for the *M. musculus* challenge. In both challenges, the distance-based DCLEAR method outperformed all other participants. In general, DCLEAR's performance in both challenges and under both metrics was excellent (Figures 2B and 2C) and although the *M. musculus* tree was ten times larger, *DCLEAR* scored higher compared with the *C. elegans* tree.

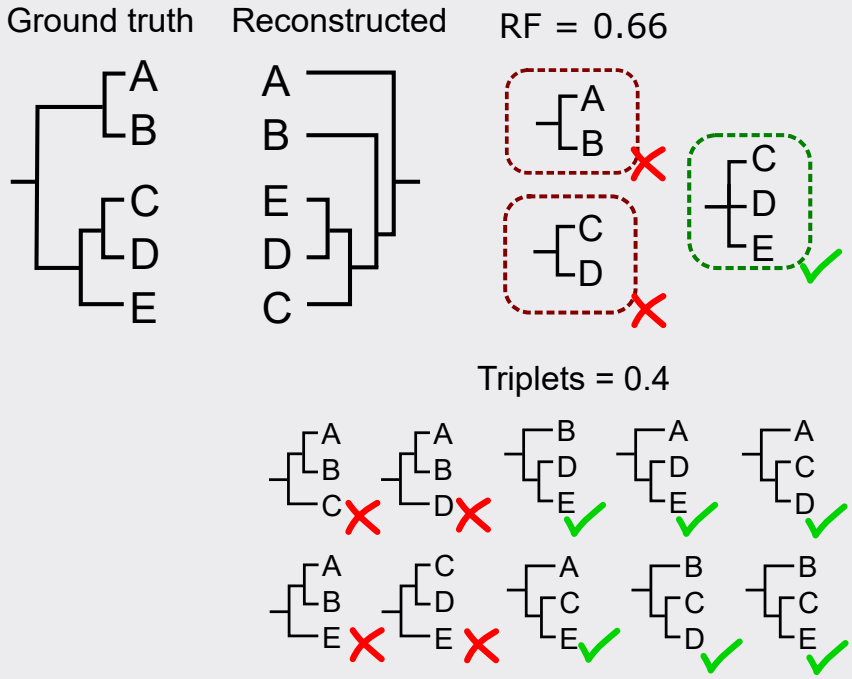
Summary statistics for the *in vitro* challenge

Given that the *in vitro* challenge predictions consisted of 30 trees of different sizes, we were able to further analyze the results. When considering only perfectly reconstructed trees, defined by a distance value of 0, AMbeRland* performed better under triplets (28 trees across teams Figure 2D, top) than under RF (21 trees Figure 2D, bottom). This discrepancy indicates that even when all triplets from a tree are correctly inferred, there might still be incorrect clades in the tree as measured by RF. We then asked whether the different teams performed better depending on the size of the tree, a main constraint for inference accuracy. Larger trees were defined as having more than 8 leaves/cells and small trees as having less or equal to 8 leaves/cells. Irrespective of the tree size, AMbeRland* also performed better (see Figure 2E). To visualize that indeed tree size has an overall effect on reconstruction accuracy, we plotted the accuracy of individual trees in both metrics colored by the number of cells per tree (Figure 2F). Across all trees and submissions, the two metrics correlation is overall high ($r = 0.77$), but it becomes clear that larger trees generally have a larger triplet distance compared with RF. A total of six trees were reconstructed perfectly by at least one of the teams (Figure S4), and we noted that these perfect trees consisted of small trees of less than 9 cells. For these small trees, edit patterns can be slightly redundant without affecting accuracy (e.g., tree 1 in Figure S4) indicating that the size of the tree is a dominant factor in reconstruction accuracy. The largest perfect tree (tree 20, Figure S4) comprises 9 cells with redundant mutations in two array states across cells, despite this, the tree can still be perfectly resolved. More generally, higher redundancy in array states effectively decreases the information that can be used for lineage reconstruction, and we indeed observed high levels of redundancy in several trees with an average of 65% ± 20% of cell arrays being unique

Box 2. Scoring approach

We applied two widely used metrics for tree comparison: the Robinson-Foulds distance and the triplets distance. While both metrics are applied to assess tree similarities, there is no clear agreement as to which one is more relevant for lineage trees. We decided to use both metrics as a way of evaluating their correlation and the insight they provide about the lineage relationships. The Robinson-Foulds distance is commonly defined as the number of partitions shared by a pair of trees across all possible partitions. A partition refers to any cut in the internal branches of a tree that would generate two sub-trees containing complementary leaves. Since the ground truth and the inferred lineage contain in total the same set of leaves, we can define a shared partition if there is a way to cut both the inferred and ground-truth trees such that the resulting sub-trees share the same sets of leaves. We obtain the RF distance by normalizing to the maximum possible distance of 1, when there are no shared partitions by the trees (Robinson and Foulds, 1981). On the other hand, the triplet distance enumerates all possible combinations of three leaves and their corresponding lineage relationship in both the ground truth and the inferred trees. One then counts the number of shared triplets and normalizes by the total possible number of triplets to obtain the triplet distance. For both metrics, a distance value of 0 means that the ground truth and inference trees are identical under the specific criteria while a distance value of 1 means that the inference is comparable to a random guess on the tree structure. Overall, the Robinson-Foulds metric detects main branching events, while the triplet metric is a better measure of local branching events.

We here present an illustrative example with left the ground truth and right the predicted tree. In this case, the tree has three possible partitions top right and ten possible triplets bottom left. Since 1 out of three partitions was incorrect the RF distance is 1/3 or 0.66. Similarly, 4 out of 10 triplets were incorrect for a triplet distance of 4/10 or 0.4. Higher distance implies more differences between the ground truth and the inference and therefore a lower score. As observed in the results of this challenge, the relationship between the two metrics will depend on the tree topology but also on the tree size. Indeed, the number of triplets will size as the cube of the number of nodes, while the RF partitions will scale linearly with the number of nodes.



(Table S1). However, tree reconstruction was not affected by this (Figures S3D and S3E). Considering non-perfect trees, the largest tree with the highest score was reconstructed by AMbeRland (29 leaves/cells, 55% unique arrays RF distance = 0.44 and triplet distance = 0.40, Figure S5). The second largest tree with high score was reconstructed by *Cassiopeia* (23 leaves/cells, 71% unique arrays, RF = 0.48, triplets = 0.70, Figure S5). In tree 29 we noted that some cells with identical array states were placed correctly in the reconstruction, this is due to the fact that AMbeRland* and *Jasper06* decided to leverage the biological restriction that lineage trees must be binary. Therefore, they imposed a binary structure even when cells had identical array states, reaching slightly higher accuracy (Figure S5).

Methods summary

The best performing methods across challenges can be roughly divided into three groups: (1) distance-based methods such as the best performers Liu's method, Guan's method, and

CellPress
OPEN ACCESS

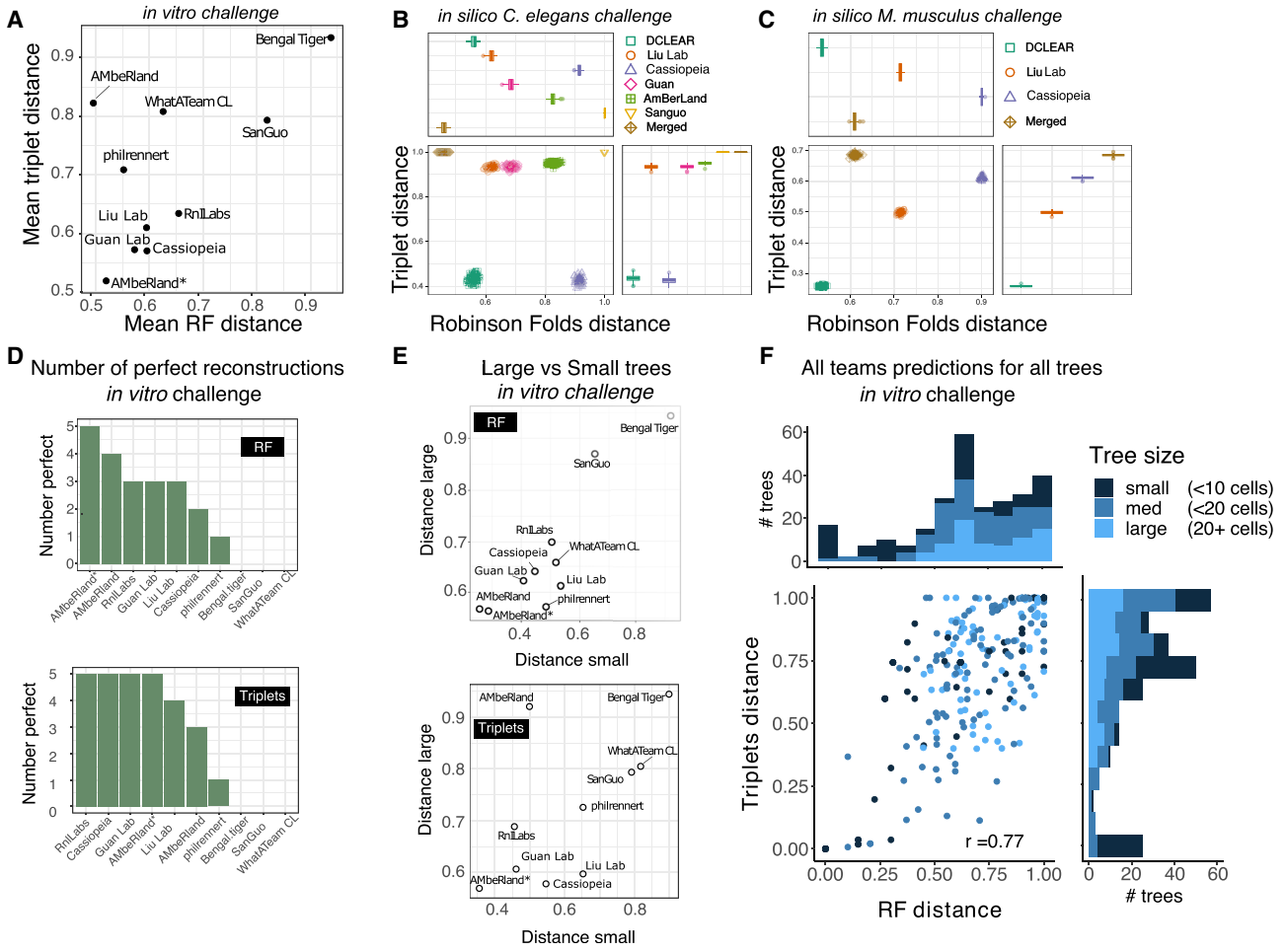


Figure 2. Analysis of challenge results

(A) Average performance across 30 lineages of all teams by both triplets and RF metrics for the *in vitro* challenge.

(B) Average bootstrapped performance of all teams by both triplets and RF metrics for the *C. elegans in silico* challenge. Error bars represent standard deviation across the bootstrapped trees where 10% of the leaves were randomly ablated.

(C) Average bootstrapped performance of all teams by both triplets and RF metrics for the *in silico M. musculus* challenge. Error bars represent standard deviation across the bootstrapped trees where 10% of the leaves were randomly ablated.

(D) Number of perfectly reconstructed lineages for each team in the *in vitro* challenge.

(E) We partitioned the *in vitro* challenge test data into large (more than 8 cells) and small (less or equal to 8 cells) trees, to assess performance by tree size.

(F) The scores for the two metrics of all 30 trees for all 9 teams for the *in vitro* challenge are plotted against each other and color coded depending on the size of the tree. Deep blue dots, small trees (total cells < 10), gray blue dots (10 < total cells < 20), light blue dots (total cells >20). Scores show a general correlation (r = 0.77) between the two metrics but also significant dispersion especially for larger trees.

DCLEAR, (2) a machine-learning-based method to predict probabilities of sister cells using a gradient boosting machine AMbeRland, and (3) a maximum-parsimony-based method Cassiopeia-integer linear programming (ILP) and Cassiopeia-Greedy. The distance-based methods reconstruct the lineage trees by first defining a distance to build a matrix between all pairs of cells as the distance between mutated characters in two cells' arrays should be proportional to the time since they split from a common ancestor. Therefore, distance matrices are commonly used in phylogenetic inference and clustering (Jones et al., 2020) or by hierarchical algorithms that represent the distance matrix as a tree such as in neighbor joining (NJ) (Saitou and Nei, 1987). Conversely, the machine-learning approach learns from the training set the importance of features/mutations to predict whether two cells are sisters. *Cassiopeia*'s maximum parsimony method reconstructed trees by minimizing the total number of steps required to explain a given configuration of the leaves. Distance-based methods combined with hierarchical clustering overall performed well with the additional advantage of being scalable. Hamming distance is a metric used for phylogenetic analysis where the distance between sequences from two taxa (or cells in this case) is calculated as the number of different sites between the two sequences. While in the traditional Hamming distance, every mutation is assigned the same weight, in lineage recording technologies the editing rates of each array character are generally not uniform (Figure 3A; Box 1), and so, mutations that occur with higher frequency are likely to arise independently in non-related cells, confounding the

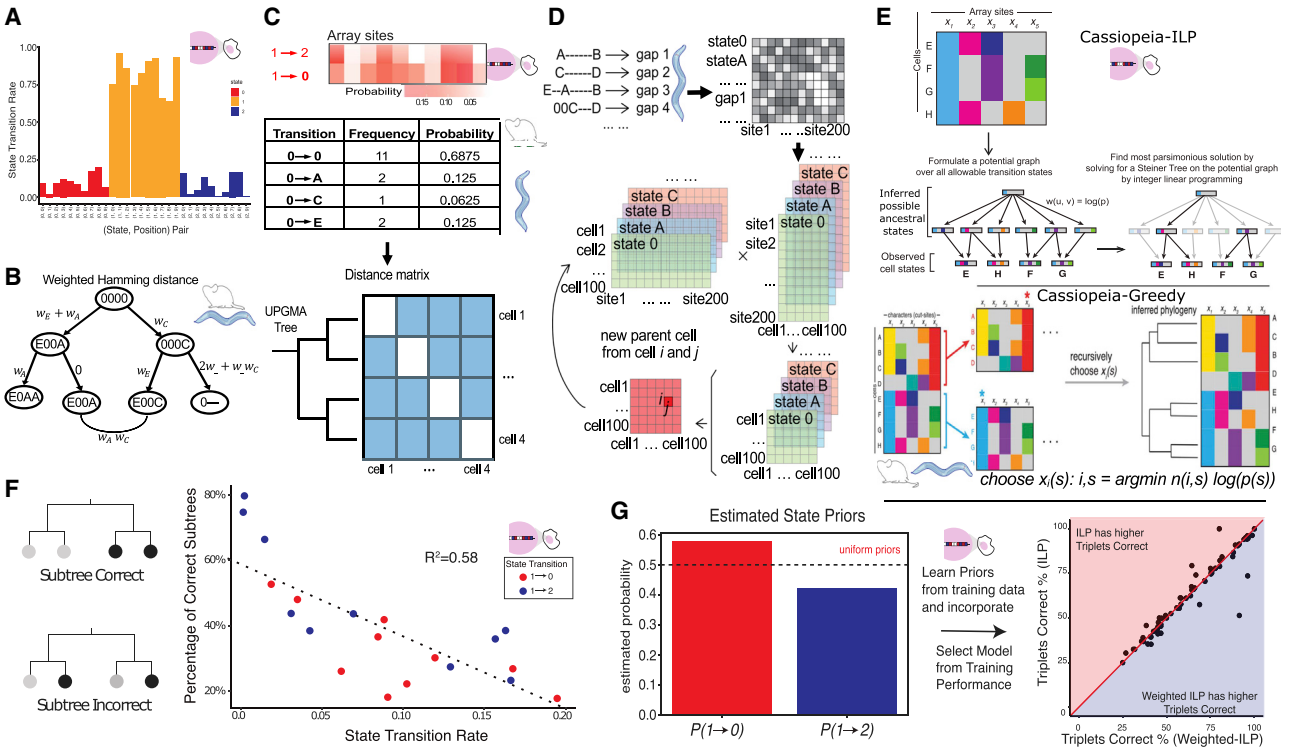


Figure 3. Different approaches for solving lineage trees and using the training data

(A) In the *in vitro* challenge, the transition rates from the unedited state (1) to either of the two edited states (0, 2) can be learned directly from the training data, the probabilities for all possible transitions at each of the ten array positions are shown as extracted from the training set.

(B) The schematic shows that when computing the sequence distances, instead of assigning equal weight to different character replacement as in Hamming distance, the weighted Hamming distance (WHD) assigns different weights to different character replacements.

(C) Description of Liu lab's method in all 3 challenges. First, for the *in vitro* challenge, the transition probability is calculated by counting the frequency of every state transition from parent node to child node. For the *in silico* challenges the transition probability for all character arrays is extracted. Next, the pairwise cell distance is defined as the likelihood of two cells' states arising from two independent events. Finally, the cell lineage is reconstructed from the distance matrix using the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) method. (D) This schematic shows the *Guan Lab*'s method used to reconstruct the *C. elegans* tree. First, all gap mutations are remarked based on mutation types at both ends, since gaps, even at the same sites, could be the results of different mutation incidents from simultaneous mutations at both ends. Then the mutation weights are generated for each mutation state at each of the 200 sites in the array and are given by $1 - \log10(p)$, where p is the observed probability of the mutation on that site. The weights define how important characters should be considered when comparing the mutation states between cells. Then bifurcate clustering of nearest cells was carried out based on matrix calibration. In the training set, the characters of all cells at all sites will be presented as n 200 by 100 matrices, where $n = 30$ is the number of array characters (0, A, B,...). The inner product of the matrices, which is n 100 by 100 matrices, reveals the relationship between the 100 cells in each tree of the training set themselves according to the 200 states, and the sum of n product matrices gives the overall pairwise similarity relationship of the 100 cells, where we can extract the most similar cell pair by the maximum value in that matrix (denoted as dark red, and the indices of the cells are denoted as I and j). Then a parent cell, generated based on the shared mutations of the two cells, replaces the two cells and is sent back to next iterations of bifurcate clustering, until only one pair of cells is left and their parent cell will become the tree root.

(E) Top: for the *in vitro* challenge Cassiopeia-ILP (Yosef Lab) takes as input a "character matrix," summarizing the mutations seen at heritable target sites across cells and infers a Steiner tree, finding the tree of minimum weight connecting all observed cell states across all possible evolutionary histories using integer linear programming (ILP). Importantly, the edges connecting cell states can be weighted by the number of mutations along that edge or the log likelihood of these mutations. Bottom: for both *in silico* challenges Cassiopeia-Greedy infers a phylogeny from the observed character states across all cells, which can be summarized in a cell's x cut-site "character matrix." To do so, the algorithm recursively applies a heuristic to split cells into two groups based on the frequency of a given state at a character, $n(I, S)$, and the likelihood of that state arising, $p(s)$. This procedure is applied until a full phylogeny is resolved.

(F) Using the 76 trees in the training set of the *in vitro* challenge to compare the relationships between cells that share a particular state, Liu lab quantified how rarer states are more predictive of the true relationship between pairs of cells. As observed in the plot, these relative rates can vary by both identity and for each of the ten positions in the target array.

(G) Cassiopeia-ILP (Yosef Lab) is able to incorporate learned state priors by weighting evolutionary transitions by their log-likelihoods and find a weighted parsimony solution. Performance on the training data can inform whether weighted or unweighted parsimony is better suited.

analysis. Conversely, some edit patterns are unlikely to happen independently and could be informative of a true inheritance event. Therefore, the uneven frequency of array edits suggests that each array element could potentially bring different information about the underlying lineage relations. To calculate the weighted Hamming distances between cells, several teams transformed the initial edited array sites of all cells in the lineage to their observed mutation frequencies and calculated the

absolute difference between the arrays of two cells (Figure 3B). Tables S2 and S3 include a concise summary of all methods. For the *in vitro* challenge we included the type of parameters or features that different teams estimated from the data, how the tree built from their estimations was and how they used the training dataset to estimate or learn the different features and parameters (Table S3). For the *in silico* challenges, given the larger scale of the trees, we also show the CPU running time as well as the code accessibility (Table S3).

Liu: Inference of internal states

In all three challenges team Liu's method reconstructed internal nodes to represent the ancestral nodes that likely gave rise to the leave cells. For the *in vitro* challenge, the state of every internal node is inferred using the states of its children by applying the following rule for each site: the parent node gets the state of the children nodes if both children states are the same, alternatively it gets the unedited state if its two children states are different. Next, for each array element, the transition rate from state "1" to state "0" or "2" is calculated as the probability of parent node having state 1 and child node having the mutated state (Figure 3C, top). Finally, the pairwise distance between two cells is considered to be the probability of two cell states arising from independent events, that is, the product of the transition rate of shared states between the two cells. In a similar way for the *in silico* challenges, team Liu estimated the character array of the internal nodes based on the fact that a target can only mutate once (Figure 3C, middle). Deletions or dropouts were replaced by the initial character "0." After inferring all the internal nodes, Liu's method derived the empirical transition probability from the ground state to the 30 possible mutated states, "A–Z" and "a–c," or deletion "-." This empirical distribution was then used to calculate the probability of two cells arising from two independent events, assuming that each target was independent of the other. The log likelihood of the transition probability for shared states was considered as the cell-to-cell distance. Finally, the distance matrix was clustered using unweighted pair group method with arithmetic mean algorithm (UPGMA) (Figure 3C, bottom). For the *M. musculus* challenge Liu's method added an extra step for clustering taking into consideration the 11 different types of cells.

Guan: Weighted Hamming distance

For the *in vitro* challenge, Guan Lab's method first designed a rule-based hierarchical clustering method using weighted Hamming distances (WHD) between cells (Figure S6A for frequency and weight values). Guan Lab transformed the initial edited array sites of all cells in the lineage to their observed mutation frequencies while retaining the mutation directions by mathematical signs (± see Figure S6A) and calculated the weighted distance as the absolute difference between the arrays of two cells. Finally, the lineage was reconstructed using a rule-based hierarchical clustering method (Figure S6B). For the *C. elegans* challenge they first replaced all gap mutations with the mutation types at both ends, since gaps even at the same sites could be the result of simultaneous mutation incidents (Figure 3D). The mutation weights were defined for each of the 200 characters in the *C. elegans* array as $1 - \log_{10}(P)$, where P is the observed probability of the mutation at that site. An iterative bifurcate clustering process was performed to combine the nearest cells based on matrix calibration, until there was only one pair of cells left and their parent cell was defined as the root of the tree (see Figure 3D).

Cassiopeia: Combinatorial optimization

Yosef Lab was the only team that did not opt for hierarchical clustering but instead, they used combinatorial optimization. For the *in vitro* challenge, the team adapted the previously published Cassiopeia-ILP (Jones et al., 2020) approach that takes as input a "character matrix," summarizing the mutations seen at heritable target sites across cells, and infers a Steiner Tree which corresponds to the maximum parsimony tree connecting all observed cell states across all possible ancestral states' histories (Figure 3E, top). The Steiner Tree is found using an integer linear programming (ILP) optimization procedure that maximizes the parsimony over all possible trajectories that could have generated the observed barcode states and consistently finds a near-optimal solution. Importantly, the edges connecting cell states can be weighted by the number of mutations along that edge or the log likelihood of these mutations. A derived method Cassiopeia-Greedy was implemented for the *C. elegans* and *M. musculus* challenges adapting a different maximum parsimony-based strategy to infer the phylogeny from a set of observed character states across all cells summarized in the "character matrix" (Jones et al., 2020). To do so, the algorithm recursively applies a heuristic to split cells into two groups based on the frequency of a given state at a character and the likelihood of that state arising, taking into account mutations that occurred earlier in the tree (Figure 3E, bottom). This procedure was applied until a full lineage tree was resolved.

Usage of the ground truth

For the *in vitro* challenge, several teams computed the calculated transition rates across the 76 trees in the training data and found striking variability across the array element identities and positions (Figure 3A). It is possible to assess in several ways how much information regarding the correct lineage of a cell is contained in the transition rate of a particular mutation. For example, given a tree in the training set it is possible to assess whether cells having the same mutation in an array element are in the same subtree branch (see diagram Figure 3F). To obtain the percentage of correct branch positioning associated to this mutation, this process can be repeated for all trees. It can then be expanded to all ten elements in the arrays, and for the two types of mutations (1 to 0 or 1 to 2). This information was used to quantify how for a given mutation and array position there is a negative correlation between the state transition rate and how well it can establish the correct relationships between four cells in a subtree ($R^2 = 0.58$, see plot Figure 3F). This observation is in line with teams assigning the observed mutation frequencies to the Hamming distance weights of different array elements but also shows that weight values can be further refined when training data are available.

Participant teams used this type of information differently as Cassiopeia-ILP (Yosef Lab) used the average across sites of the transition probabilities for each type of mutation to weight the edges of their Steiner Tree search (Figure 3E, top).

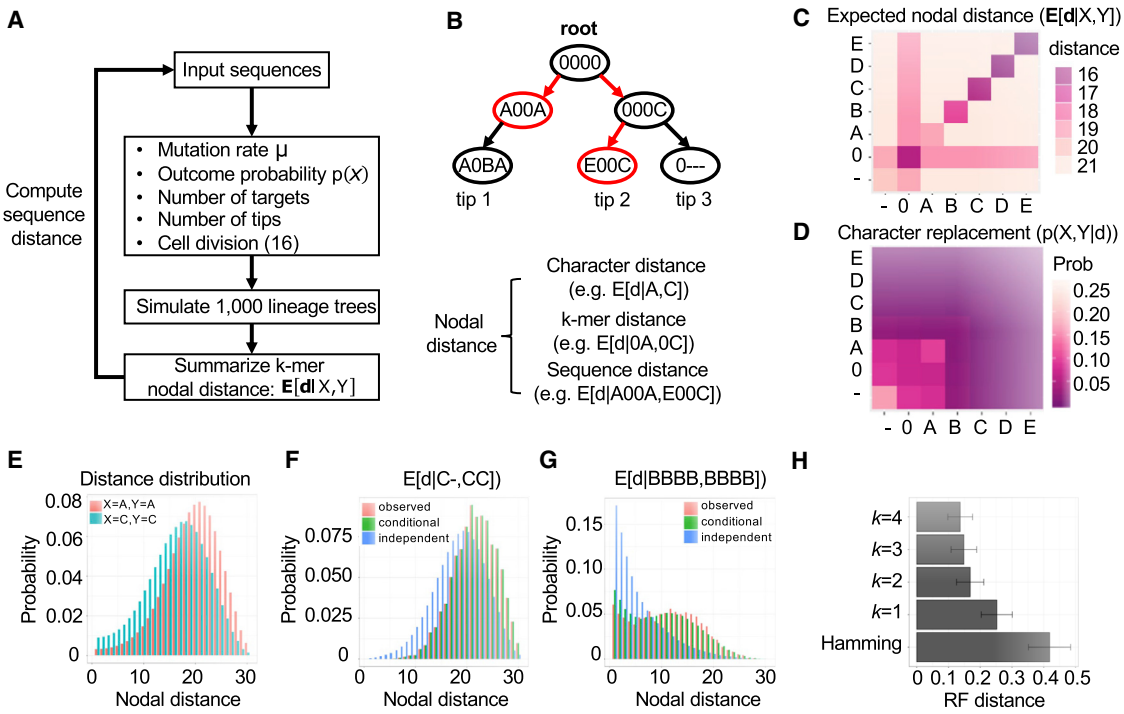


Figure 4. DCLEAR learning k-mer replacement distances by simulation

(A) The input sequences were first used to estimate the summary statistics such as mutation rate (M), outcome probability of each character, number of targets and number of tips. These estimated parameters, combined with the pre-defined parameters such as cell divisions, were used to simulate multiple lineage trees from the root node. The k-mer nodal distances were estimated from these simulated lineage trees and then used to compute the distances between input sequences.

(B) The schematic shows a simulated lineage tree with one root, two internal nodes and three tips. The nodal distance is defined as the distance between any two nodes on the lineage tree. The expected nodal distance can be estimated from the replacement of individual characters (e.g., between A and C), the replacement of k-mers (e.g., between 0A and 0C), or sequences (e.g., between A000A and E00C).

(C) The heatmap shows the expected nodal distance of the replacement of the most frequent individual characters.

(D) The heatmap shows the probability of replacement of the most frequent individual characters at a nodal distance of 15.

(E) The histogram shows the posterior distribution of nodal distance of two sequences when having the same characters (A) or (C) at any specific position.

(F and G) The histograms show the observed distribution (red bars) and estimated posterior distribution of nodal distance of two sequences (F) with the replacement of C- by CC, or (G) with BBBB at the same position. The posterior distributions were estimated by using an independent model (blue bars) and a conditional model (green bars). In both cases, the posterior distribution estimated by the conditional model is more consistent with the observed distribution.

(H) The simulated trees were used to compare the performance of lineage reconstruction by using Hamming distance and k-mer replacement distances with different k's. We simulated 1,000 lineage trees with cell division of 16, mutation probability of 0.1, 200 targets and 200 tips. The outcome probability was sampled from a Gamma distribution with shape of 0.1 and rate of 2. For both k-mer replacement distances and Hamming distance, we used a balanced minimum evolution (ME) algorithm with tree rearrangement (nearest neighbor interchange, subtree pruning and regrafting, and tree bisection and reconnection) to infer the tree topology. The similarity between 1,000 inferred and simulated trees was measured by the Robinson-Foulds (RF) distance, error bars represent standard deviation across all trees.

Additionally, for this team the training data also proved useful in choosing a model as they were able to compare the performance of different algorithms and select the one that performed the best (Figure 3G). Team Guan Lab was able to use the ground truth for comparing several types of distance-based tree construction methods, including NJ and UPGMA. This analysis showed that UPMGA performed similar to their rule-based hierarchical clustering, whereas NJ was significantly outperformed (Figure S6C). Finally, DCLEAR(WHD) used the training set to weight the mutations for the *C. elegans* tree and AMbeRland used a gradient boosting machine (GBM) to learn the relative importance of several features derived from the array states data and for determining the clustering thresholds for the tree reconstruction (see details below).

DCLEAR estimates k-mer replacement distances by simulation

DCLEAR (distance-based cell lineage reconstruction) implemented two best performing strategies to compute the cell distances, a WHD that requires a training set for optimizing each mutation weight for the *C. elegans* tree, and a k-mer replacement distance (KRD), that does not require training data, for the *M. musculus* tree. *DCLEAR* (KRD) first looks at mutations in the character arrays to estimate the parameters of the generative process associated with the tree to be reconstructed. With these parameters, they repetitively simulated trees with a size and mutation distribution similar to the *M. musculus* target tree (Figure 4A). The k-mer replacement distances were estimated from the simulated lineage trees and used to compute the distances

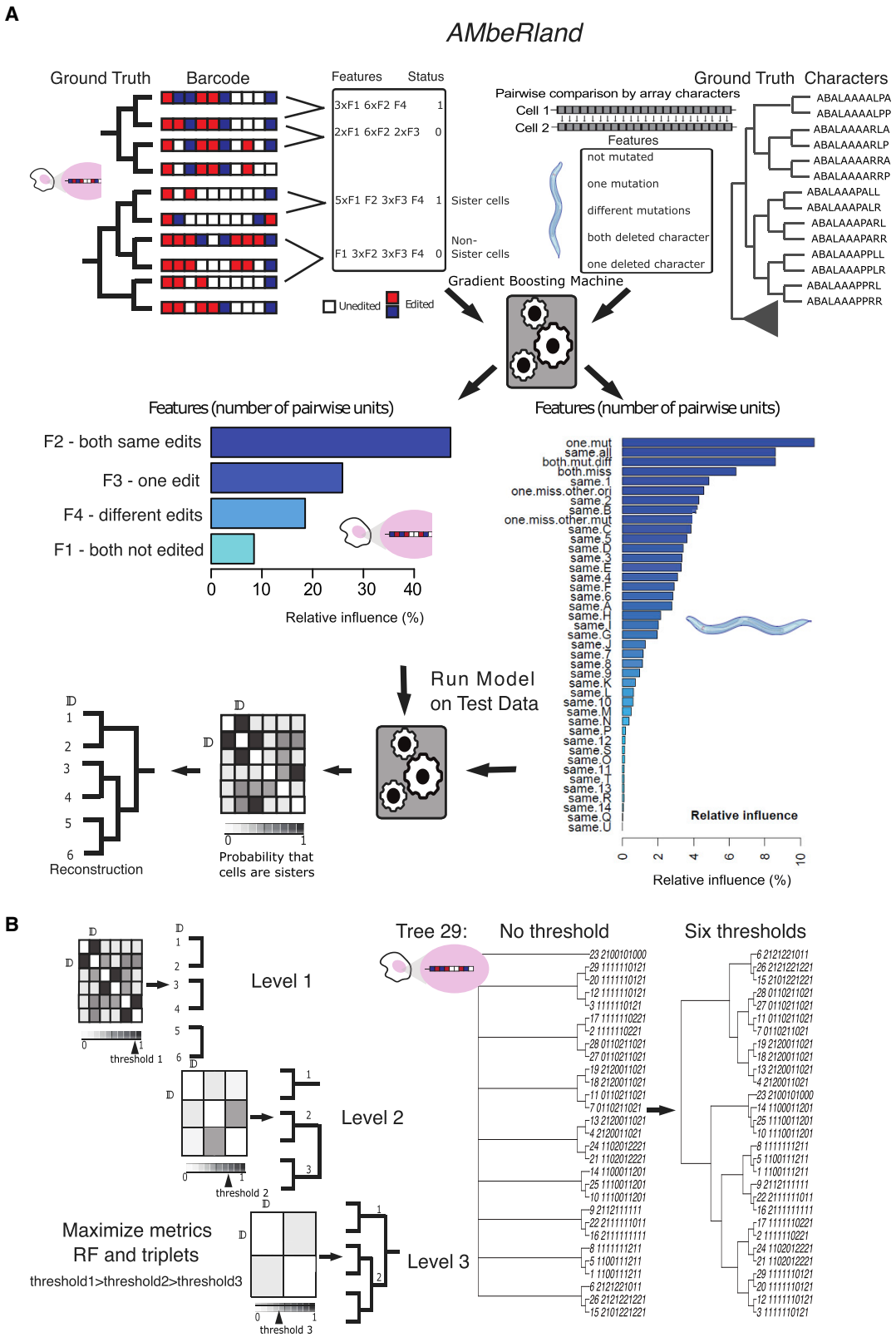


Figure 5. AMbeRland: A decision-tree-based approach for reconstruct cell lineages

(A) After selecting manually different model features for left the *in vitro* challenge (F1 to F4) and right the *C. elegans* challenge, AMbeRland learns the features importance represented by histograms of the weights, for predicting phylogenetic relationships directly from the training data using a gradient boosting machine

(legend continued on next page)

between input sequences in the character arrays of internal nodes and tips. As a toy example, two cells in a simulated tree have respectively the character arrays A00A and E00C, their *1*-mer nodal distance will be the distance between A and C, their *2*-mer nodal distance will be the distance between 0A and 0C while the whole sequence nodal distance will be between A00A and E00C (see red cells in Figure 4B). Specifically, by examining the simulated lineage trees, DCLEAR (KRD) estimated the expected *1*-mer replacement distance between characters in the array (including ground state "0" and deletion state "-") in the lineage trees (Figure 4C) and the probability for a given nodal distance of replacing a character in a cell array (Figures 4D and 4E). To extend the *1*-mer replacement distance to the KRD, the posterior probability distributions of KRD were estimated by using a conditional model considering a dependance for the concurrence of mutations (Figures 4F and 4G). They found that by considering the neighboring characters, the conditional model can more accurately estimate the nodal distance than an independent *1*-mer model. The cell distance can then be readily computed as the mean expected KRD (see STAR Methods). Similar to WHD, the lineage trees were reconstructed using the minimum evolution (FastME) or NJ algorithms (Gascuel and Steel, 2006; Lefort et al., 2015). For both DCLEAR WHD and KRD, the deletions and dropouts were treated differently. In WHD, the weight for deletion, dropout, regular state, and ground state are 0.9, 0.4, 3, and 1, respectively. In KRD, deletion and dropout are treated as two different characters.

AMbeRland, a decision-tree-based method

AMbeRland's approach relied on machine learning to build a distance matrix between cells through the calculation of the relative importance of features derived from the states of the character arrays (Figure 5). In their approach for the *in vitro* challenge, they first defined four features for every pair of cells consisting of whether two cells are both unedited at a given array site (feature F1), a site has the same edits (feature F2), only one site is edited (feature F3), or if both sites have different edits (feature F4) (Figure 5A, left). Then, the prevalence of these four features was extracted for a group of ~500 pairs of sister cells (label 1) and ~3,000 non-sister cells (label 0) using the 76 ground-truth trees available in the training set. Finally, gradient boosting (Friedman, 2001) was applied to learn from these data the relative weights of each feature to predict whether two cells are actually sisters (see Figure S7**)**. For the *C. elegans* challenge AMbeRland applied a similar approach using the training set of 100 trees with 100 cells. They similarly determined weights for features selected by counting pairwise positions in two cell's arrays that were (1) not mutated, (2) had a single mutation, (3) both had different mutations, (4) both had a missing record, (5) one had a missing record and the other not mutated, etc. (Figure 5A, right).

In both challenges, AMbeRland applied a custom hierarchical clustering method for building the cell lineage tree from the predicted probabilities. During the tree construction, the ground truth was used to evaluate a set of decreasing thresholds corresponding to how any two individual clusters of cells were related at different levels of the lineage tree (see Figure 5B, left). The clustering starts at the lowest tree level, where all cell pairs are ordered according to the predicted probability that they are sister cells, from here, cells with a probability higher than the first threshold are assigned as pairs, while the rest are kept as a branch with a single cell. At each consecutive level, pairwise comparisons are performed between each lower level cluster by calculating the maximum probability between any two elements of the two clusters. Pairs of clusters were ordered again according to this probability and were assumed to have the same parent node if their value was above the estimated threshold for this level. This process was repeated until one or two clusters were left. The values for the thresholds at each level were determined by performing a grid search minimizing the RF and triplet distance metrics (see results for tree 29 Figure 5B, right). This procedure clearly helped obtaining better scores, particularly regarding the triplet metric (see Figure S8 for all trees in the *in vitro* challenge and Figure S9 for *C. elegans*).

Consensus trees

One advantage of having a set of different and diverse approaches trying to solve a common problem is that it is possible to aggregate the solutions and gather collective insight. Hence, we decided to test how a consensus tree of all teams would perform compared with individual methods (Figures 6A and 6D). For the *in vitro* challenge, we constructed the consensus tree using the submissions from all teams (excluding Bengal Tiger because of their unusual number of low-accuracy outliers, Figure 2A) by applying the majority rule algorithm (Felsenstein, 1985). Interestingly, we see that the consensus tree performs better than any individual team when considering the RF distance, but this is not the case according to the triplet distance (Figure 6B). To further understand this, we evaluated the agreement (or support) of each clade in a given tree across teams using the Felsenstein's bootstrap proportion (FBP), which has been traditionally used to assess the support of phylogenetic trees (Felsenstein, 1985). For FBP agreement, a branch must match a reference branch exactly to be accounted for in the score, so we define FBP as a strict agreement (Figure 6A). Alternatively, the transfer bootstrap expectation (TBE) provides higher resolution estimates of branch support and can be used to assess phylogenetic similarity even when there is no strict majority consensus (Lemoine et al., 2018). The distribution of FBP and TBE support scores at different normalized depths across all 30 trees in the test dataset shows that the inference of earlier clades varies significantly across methods, whereas late splits are resolved correctly by the majority (Figures 6A and S10). The divergence for earlier clades might explain the lower performance of the consensus tree under the triplet metric, given that for these small trees more triplets are prone to include early divisions with wrong clade relationships (see Figures S3A and S3B).

(GBM), middle. These learned weights are then used to predict the probability of sister-cell relationships on the hold out test data creating a probability matrix used for hierarchical reconstruction, bottom.

(B) Left: trees are reconstructed from probability matrices by performing a grid search to obtain the clustering thresholds at each tree level while maximizing the RF and triplets metrics. Right: example of differences when establishing thresholds for tree 29, the largest correctly reconstructed tree in the *in vitro* challenge. See also detailed examples in Figures S7 and S8.

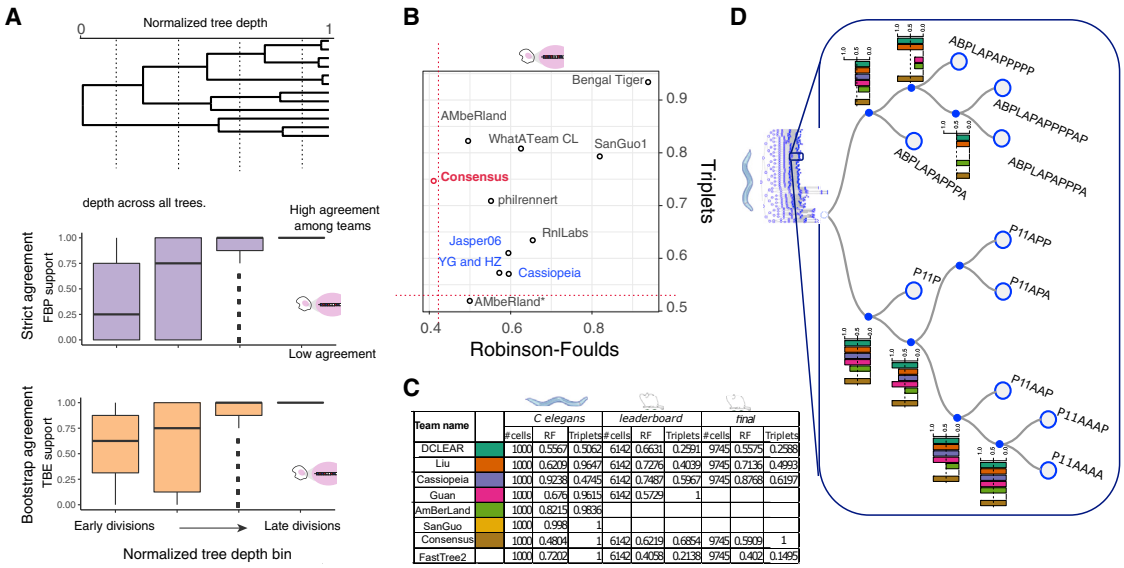


Figure 6. Consensus methods and agreement in tree reconstruction

(A) Depth-dependent agreement between reconstructed trees calculated by Felsenstein bootstrap Proportion and transfer bootstrap expectation. Both metrics assess the degree of agreement that different trees have on specific splits (or cell divisions). High agreement indicates that most teams resolved splits correctly at that depth. The distribution is computed across all 30 trees in the *in vitro* test sets. Error bars represent standard deviation across all trees partitioned by size. (B) We computed the consensus trees by majority rule using the *consensus* function from the *R package* **ape** v5.3. The consensus performance in the *in vitro* challenge is higher than any individual team by RF distance but not by triplets (red dotted line indicates the best performed by each metric). (C) Scores summarizing all participating methods for the *in silico* challenges, including the PHYLIP (Phylogeny Inference Package) consensus and for reference *FastTree2*. (D) Annotated subtree of *C. elegans* challenge, edges are marked with tables listing the agreement of each of the 5 individual submissions and the consensus in transfer bootstrap distance where 1 is high agreement. Colors refer to the table in (C).

For the *in silico* challenges, we also added for comparison the performance of the algorithm *FastTree2*, a fast and reliable approximately maximum-likelihood method (Price et al., 2010) that performed better than NJ or TripleMaxCut (Sevillya et al., 2016). Interestingly, we observed that in the *C. elegans* challenge, DCLEAR outperforms Fastree2 by both metrics, which is not the case for the *M. musculus* challenge as FastTree2 outperforms all methods, with DCLEAR as a close second (Figure 6C). We also see that for the *C. elegans* challenge, the consensus tree performs better than any individual team when considering the RF distance, but under the triplet distance the consensus is nevertheless equivalent to a random submission (Figure 6C). In the *M. musculus* challenge there were probably not enough submissions to see a "wisdom of the crowds" effect as the consensus tree does not outperform DCLEAR. To understand the difference between the RF and triplet distances, we evaluated the agreement of each clade in the *C. elegans* tree across teams. Overall, as in the *in vitro* challenge we observed a depth-dependent effect in the support between teams, as measured by TBE (Figure 6D) and the divergence for earlier clades might explain the lower triplet metric performance in the consensus tree solution but in this case probably due to the *C. elegans* tree topology having many internal nodes.

DISCUSSION

The main goal of this DREAM challenge was to mobilize a larger community to generate new methods for cell lineage reconstruction. This goal was catalyzed through the generation of new *in silico* datasets and by the recent availability of *in vitro* datasets with an associated ground truth. This study represents the first attempt to rigorously examine the performance of various algorithms across diverse molecular tools and lineage trees. For the *in vitro* challenge a total of nine approaches were submitted for which the maximum performance plateaued (see Figure 2A; Table S2). While some trees were reconstructed perfectly, the scores were far from the theoretical maximum. We thought this could be mainly due to the high degeneracy in cell arrays where two or more cells show identical edit patterns, but further analysis showed that bar-code degeneration did not affect the performance of the teams (Figure S3E). This problem could be in principle overcome by increasing the memory of the intMEMOIR system, as discussed by the authors (Chow et al., 2021). On the other hand, the degeneracy problem was non-existent for the *C. elegans* tree as all cells ended up with a different mutational character array and was minimal for *M. musculus* with only ~2.7% of sister cells sharing exactly the same character arrays. Indeed, the choice of the mutation rate and the diversity of mutations in the simulations has a strong effect on the accuracy of cell lineage reconstruction as low diversity of possible mutational outcomes generally gives poorer results. While too low mutation rates lead to more unedited and therefore non-informative targets, too high mutation rates lead to most targets being mutated during the early cell divisions, leaving few targets available for recording later events (Salvador-Martínez et al., 2019). Hence, we tuned our *in silico* mutation rates and array sizes in

order to avoid cells having identical character arrays. As the performance of *DCLEAR* in the *in silico* challenges was as good or even better than the results of the *in vitro* challenge (see Figure 2), the limits of its performance must derive from the tree size or topology. We concluded that tree topology was the most important parameter given that DCLEAR *M. musculus* lineage reconstruction was more accurate than for the ten times smaller *C. elegans* tree. Given these great performances, we also consider the *in silico* challenges as a success despite not having as many submissions, as the diversity and performance of the approaches was impressive (see Figure 2; Table S3).

The implementation of several metrics to evaluate the participants was also an original feature of the challenge as typically, lineage trees are evaluated with a single metric, and no comparison between metrics is systematically performed (Salvador-Martínez et al., 2019). This aspect was essential not only to thoroughly evaluate participants (Figures 2, S2, and S3) but also to better understand their solutions. One of the striking observations was the disconnection in all challenges of the performance as measured by the two metrics. Indeed, for the *in vitro* challenge AMbeRland optimized post competition their algorithm for the triplet distance and had the overall best performance without compromising their RF performance (Figure 3A). Also, for larger trees, team AMbeRland had overall a similar performance than Cassiopeia relative to the triplet distance (average triplets = 0.55) but scores better in the RF metric (RF = 0.57 and 0.65 respectively, see Figure 2E). We see the opposite for team *phil-rennert* although now the difference for larger trees appears for the triplet distance (triplets = 0.57 and 0.72, respectively, see Figure 2E) as the RF distance is similar. Such dissociation between metrics was also observed for the majority-vote consensus solution, which had the best score for RF but far from that for triplet distance (Figure 6B). The analysis of the overall agreement between individual solutions at different depths of the trees shows that indeed for earlier cell divisions agreement is low (Figure 6A). This observation provides a possible explanation for the divergence between triplet and RF distances, as in smaller trees such as the ones in the *in vitro* challenge, more triplets are prone to include early divisions with wrong clade relationships, bringing down the triplet performance. AMbeRland was probably able to correct this by performing a grid search and changing the thresholds for hierarchical clustering at higher levels of the tree. As AMbeRland was also the method that most consistently predicted smaller and larger trees (Figures 2D, S4, and S5), this also explains why we observed that overall the triplet distance is higher than RF in larger trees as opposed to smaller trees (Figures 2F and S3C).

For the largest trees, in the *in silico* challenges the interpretation of the metrics is different as the number of possible triplets grows cubically with the size of the tree, while the number of partitions considered by the RF distance grows linearly. Hence, for larger trees, the triplet distance will be dominated by the higher number of triplets closer to the tree leaves whereas the RF distance will be mostly measuring major branching events in the early cell division stages. As DCLEAR was consistently better in both metrics but scored less favorably in RF distance compared with the triplet distance, this suggests that DCLEAR is precisely having trouble detecting those major branching events. Indeed, both WHD and KRD in DCLEAR methods rely on rare mutations to estimate the cell distances. During early cell division stages, however, the rare mutations are significantly less likely to be present in the sequences and result in difficulties for separating early branching events. Modeling the dependence between multiple non-adjacent mutations in the sequences, on top of the neighboring *k-mers*, may be necessary to more accurately evaluate the early branching events. It is also striking to see how the maximum parsimony approach of Cassiopeia scored much better for the triplet distance for larger trees in all challenges. Finally, the machine-learning approach derived from the one applied in the *C. elegans* challenge by AMbeRland was able to perform acceptably in the RF metric with much larger trees (see Figure 2B), but although the threshold optimization worked for the training set of 100 cell trees (see Figure S9), it did not do well with the triplet distance of the *C. elegans* tree probably due to the need to include many more thresholds given its ten times larger size.

The final observation regarding the metrics discrepancy is related to the performances in the training and test sets of the *C. elegans* challenge, as all teams are similar regarding the RF distance but with the exception of DCLEAR and Cassiopeia, the triplets performance is worse for the test set than in the training set (see Box 1). Conversely, for the *M. musculus* challenge their performances in the leaderboard tree of ~6,500 cells and the *M. musculus* tree of ~10,000 cells match for both metrics (Figure 6C). We conclude that when reconstructing a cell lineage tree, the results obtained with an algorithm for a training set of trees with a number of leaves an order of magnitude smaller than the test set are comparable, although the triplet distance is more unstable than the RF distance.

Regarding the generalization of the results obtained with the intMEMOIR technology, which is difficult to compare at the molecular level to the sequence-based approaches for lineage reconstruction as it also shows differences such as the absence of accidental deletions or dropouts, we think that in conjunction with the results from the *in silico* approaches, the generalizable conclusions are the necessity of having well-calibrated mutation rates to avoid too little mutations but also array degenerations, the utility of having a training set of smaller trees to optimize lineage reconstruction methods including distances and clustering, and allowing for a clear interpretation of the effect of the two different metrics with different tree sizes.

Overall, we think that the decisions taken while producing the datasets for the *in silico* challenges were the correct ones. We were able to pose a problem that we think is close enough to a biological situation and difficult enough so that the lessons learned and solutions generated can be implemented in other contexts. Indeed, it has been estimated that under ideal conditions of optimized mutation rates, uniform cell divisions and fully sequenced targets, 30 targets should be sufficient to reach a high level of accuracy for the lineage reconstruction of a tree of about 65,000 cells (Salvador-Martínez et al., 2019). In this situation 100 targets would theoretically yield almost perfect accuracy, far from the results obtained by the solutions submitted to both challenges.

Finally, as new DNA-editing-based molecular tools promise the reconstruction of single-cell lineages from complex model organisms, including the human cell lineage, an important question is whether the access to smaller trees and the molecular

data from their cell lineages could help find solutions to be implemented for larger trees of the same origin. The *M. musculus* lineage tree being the current experimental frontier for lineage reconstruction (Bowling et al., 2020; Kalhor et al., 2018), our results show that indeed, in order to obtain an accurate full cell lineage for mouse or human, it could be possible to train algorithms on smaller trees obtained from organs (Bowling et al., 2020) or *in vitro* dividing cells and these can then be implemented for building algorithms that can then be applied to the reconstruction of much larger trees. This DREAM challenge was a first attempt to rigorously examine the performance and robustness of various algorithms under the same conditions. It took advantage of the unique opportunity to use unpublished datasets of molecular and simulated character arrays. We hope that showing that machine-learning methods can indeed be successfully implemented will pave the way for other benchmarking efforts based on emerging technologies for monitoring cell lineages and the application of new algorithmic approaches but also that the approaches described here will pave the way for the solution of the mouse and human cell lineages.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Simulation of the cell lineage trees
  - Inter-target deletions
  - Sequencing dropouts
- CHALLENGE BEST PERFORMING METHODS IMPLEMENTATION DETAILS
  - DCLEAR (Distance based Cell LinEAge Reconstruction)
  - *AMbeRland:* Using Machine Learning (ML) for cell lineage reconstruction
- QUANTIFICATION AND STATISTICAL ANALYSIS
  - Data and software availability
  - Data availability
  - Scoring
  - Code for producing the *in silico* datasets

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.cels.2021.05.008.

ACKNOWLEDGMENTS

Funding: the research was funded by the Paul G. Allen Frontiers Group Prime Awarding Agency and HFSP (RGP0002/2016) to I.S.-M. and M.J.T.

AUTHOR CONTRIBUTIONS

A.A.G., I.S.-M., O.R., Y.G., Z.L., N.Y., J.S., M.J.T., E.S., M.B.E., and P.M. designed research; A.A.G., O.R., I.S.-M., W.G., J.H., H.Z., R.R., M.G.J., and P.M. analyzed data; K.-H.K.C., I.-Y.K., Al.P., Au.P., A.K., R.Z., S.R., R.W., P.R., V.G.S., N.S., A.R., T.J., R.S., J.P., L.H., and X.S. analyzed data; A.A.G., O.R., I.S., W.G., J.H., H.Z., R.R., M.G.J., and P.M., wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Worm image in Figure 1 was modified from *Caenorhabditis elegans* hermaphrodite adult-en.svg from Wikimedia Commons by K. D. Schroeder, CC-BY-SA 3.0. The schematic cell lineage of *C. elegans* in Figures 1 and 6 was generated using the cell lineage web visualization tool *CeLaVi* available at http://celavi.pro (Salvador-Martínez et al., 2020).

Received: August 21, 2020
Revised: February 1, 2021
Accepted: May 11, 2021
Published: June 18, 2021

REFERENCES

Alemany, A., Florescu, M., Baron, C.S., Peterson-Maduro, J., and Oudenaarden, A. van. (2018). Whole-organism clone tracing using single-cell sequencing. Nature *556*, 108–112.

Becattini, S., Latorre, D., Mele, F., Foglierini, M., De Gregorio, C., Cassotta, A., Fernandez, B., Kelderman, S., Schumacher, T.N., Corti, D., et al. (2015). T cell immunity. Functional heterogeneity of human memory CD4+ T cell clones primed by pathogens or vaccines. Science *347*, 400–406.

Behjati, S., Huch, M., Boxtel, R. van, Karthaus, W., Wedge, D.C., Tamuri, A.U., Martincorena, I., Petljak, M., Alexandrov, L.B., Gundem, G., et al. (2014). Genome sequencing of normal cells reveals developmental lineages and mutational processes. Nature *513*, 422–425.

Bowling, S., Sritharan, D., Osorio, F.G., Nguyen, M., Cheung, P., Rodriguez-Fraticelli, A., Patel, S., Yuan, W.-C., Fujiwara, Y., Li, B.E., et al. (2020). An engineered CRISPR-Cas9 mouse line for simultaneous readout of lineage histories and gene expression profiles in single cells. Cell *181*, 1410–1422.e27.

Chan, M.M., Smith, Z.D., Grosswendt, S., Kretzmer, H., Norman, T.M., Adamson, B., Jost, M., Quinn, J.J., Yang, D., Jones, M.G., et al. (2019). Molecular recording of mammalian embryogenesis. Nature *570*, 77–82.

Chow, K.-H.K., Budde, M.W., Granados, A.A., Cabrera, M., Yoon, S., Cho, S., Huang, T.H., Koulena, N., Frieda, K.L., Cai, L., et al. (2021). Imaging cell lineage with a synthetic digital recording system. Science *372*, eabb3099.

Evrony, G.D., Lee, E., Mehta, B.K., Benjamini, Y., Johnson, R.M., Cai, X., Yang, L., Haseley, P., Lehmann, H.S., Park, P.J., and Walsh, C.A. (2015). Cell lineage analysis in human brain using endogenous retroelements. Neuron *85*, 49–59.

Felsenstein, J. (1985). Confidence limits on phylogenies: an approach using the bootstrap. Evolution *39*, 783–791.

Frieda, K.L., Linton, J.M., Hormoz, S., Choi, J., Chow, K.-H.K., Singer, Z.S., Budde, M.W., Elowitz, M.B., and Cai, L. (2017). Synthetic recording and in situ readout of lineage information in single cells. Nature *541*, 107–111.

Friedman, J. (2001). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics *29* (5), 1189–1232. http://www.jstor.org/stable/2699986.

Frumkin, D., Wasserstrom, A., Kaplan, S., Feige, U., and Shapiro, E. (2005). Genomic variability within an organism exposes its cell lineage tree. Plos Comput. Biol. *1*, e50.

Garcia-Marques, J., Espinosa-Medina, I., Ku, K.Y., Yang, C.P., Koyama, M., Yu, H.H., and Lee, T. (2020). A programmable sequence of reporters for lineage analysis. Nat. Neurosci. *23*, 1618–1628.

Gascuel, O., and Steel, M. (2006). Neighbor-joining revealed. Mol. Biol. Evol. *23*, 1997–2000.

Jones, M.G., Khodaverdian, A., Quinn, J.J., Chan, M.M., Hussmann, J.A., Wang, R., Xu, C., Weissman, J.S., and Yosef, N. (2020). Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. Genome Biol *21*, 92.

Kalhor, R., Kalhor, K., Mejia, L., Leeper, K., Graveline, A., Mali, P., and Church, G.M. (2018). Developmental barcoding of whole mouse via homing CRISPR. Science *361*, eaat9804.

Kebschull, J.M., and Zador, A.M. (2018). Cellular barcoding: lineage tracing, screening and beyond. Nat. Methods *15*, 871–879.

Kester, L., and Oudenaarden, A. van. (2018). Single-cell transcriptomics meets lineage tracing. Cell Stem Cell *23*, 166–179.

Kretzschmar, K., and Watt, F.M. (2012). Lineage tracing. Cell *148*, 33–45.

Lefort, V., Desper, R., and Gascuel, O. (2015). FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. Mol. Biol. Evol. *32*, 2798–2800.

Lemoine, F., Domelevo Entfellner, J.B., Wilkinson, E., Correia, D., Dávila Felipe, M., De Oliveira, T., and Gascuel, O. (2018). Renewing Felsenstein's phylogenetic bootstrap in the era of big data. Nature *556*, 452–456.

Livet, J., Weissman, T.A., Kang, H., Draft, R.W., Lu, J., Bennis, R.A., Sanes, J.R., and Lichtman, J.W. (2007). Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. Nature *450*, 56–62.

Lodato, M.A., Woodworth, M.B., Lee, S., Evrony, G.D., Mehta, B.K., Karger, A., Lee, S., Chittenden, T.W., D'Gama, A.M., Cai, X., et al. (2015). Somatic mutation in single human neurons tracks developmental and transcriptional history. Science *350*, 94–98.

McKenna, A., Findlay, G.M., Gagnon, J.A., Horwitz, M.S., Schier, A.F., and Shendure, J. (2016). Whole-organism lineage tracing by combinatorial and cumulative genome editing. Science *353*, aaf7907.

McKenna, A., and Gagnon, J.A. (2019). Recording development with single cell dynamic lineage tracing. Development *146*, dev169730.

Perli, S.D., Cui, C.H., and Lu, T.K. (2016). Continuous genetic recording with self-targeting CRISPR-Cas in human cells. Science *353*, aag0511.

Price, M.N., Dehal, P.S., and Arkin, A.P. (2010). FastTree 2–approximately maximum-likelihood trees for large alignments. PLoS One *5*, e9490.

Qiu, P. (2020). Embracing the dropouts in single-cell RNA-seq analysis. Nat. Commun. *11*, 1169.

Raj, B., Wagner, D.E., McKenna, A., Pandey, S., Klein, A.M., Shendure, J., Gagnon, J.A., and Schier, A.F. (2018). Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. Nat. Biotechnol. *36*, 442–450.

Robinson, D.F., and Foulds, L.R. (1981). Comparison of phylogenetic trees. Math. Biosci. *53*, 131–147.

Saez-Rodriguez, J., Costello, J.C., Friend, S.H., Kellen, M.R., Mangravite, L., Meyer, P., Norman, T., and Stolovitzky, G. (2016). Crowdsourcing biomedical research: leveraging communities as innovation engines. Nat. Rev. Genet. *17*, 470–486.

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

Salcedo, A., Tarabichi, M., Espiritu, S.M.G., Deshwar, A.G., David, M., Wilson, N.M., Dentro, S., Wintersinger, J.A., Liu, L.Y., Ko, M., et al. (2020). A community effort to create standards for evaluating tumor subclonal reconstruction. Nat. Biotechnol. *38*, 97–107.

Salvador-Martínez, I., Grillo, M., Averof, M., and Telford, M.J. (2019). Is it possible to reconstruct an accurate cell lineage using CRISPR recorders? eLife *8*, e40292.

Salvador-Martínez, I., Grillo, M., Averof, M., and Telford, M.J. (2020). CeLaVi: an interactive cell lineage visualisation tool. Nucleic Acids Res. gkab325. https://doi.org/10.1093/nar/gkab325.

Sevillya, G., Frenkel, Z., and Snir, S. (2016). Triplet MaxCut: a new toolkit for rooted supertree. Methods Ecol. Evol. *7*, 1359–1365.

Spanjaard, B., Hu, B., Mitic, N., and Junker, J.P. (2017). Massively parallel single cell lineage tracing using CRISPR/Cas9 induced genetic scars. bioRxiv https://www.biorxiv.org/content/10.1101/205971v1.full.

Spanjaard, B., Hu, B., Mitic, N., Olivares-Chauvet, P., Janjuha, S., Ninov, N., and Junker, J.P. (2018). Simultaneous lineage tracing and cell-type identification using CRISPR–Cas9-induced genetic scars. Nat. Biotechnol. *36*, 469–473.

Spiro, A., and Shapiro, E. (2016). eSTGt: a programming and simulation environment for population dynamics. BMC Bioinformatics *17*, 187.

Sugino, K., Garcia-Marques, J., Espinosa-Medina, I., and Lee, T. (2019). Theoretical modeling on CRISPR-coded cell lineages: efficient encoding and optimal reconstruction. bioRxiv https://www.biorxiv.org/content/10.1101/538488v2.

Sulston, J.E., and Horvitz, H.R. (1977). Post-embryonic cell lineages of the nematode, Caenorhabditis elegans. Dev. Biol. *56*, 110–156.

Wagner, D.E., and Klein, A.M. (2020). Lineage tracing meets single-cell omics: opportunities and challenges. Nat. Rev. Genet. *21*, 410–427.

Weissman, T.A., and Pan, Y.A. (2015). Brainbow: new resources and emerging biological applications for multicolor genetic labeling and analysis. Genetics *199*, 293–306.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Methods and algorithms to generate and solve the images	https://github.com/Lineage-Reconstruction-DREAM-Challenge/hub/wiki	N/A

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Pablo Meyer (pmeyerr@us.ibm.com).

Materials availability

No newly generated materials are associated with this paper.

Data and code availability

- Scripts were not used to generate the figures reported in this paper.
- Any additional information required to reproduce this work is available from the lead contact.
- As indicated, source data for the challenges has been deposited at synapse.org and are publicly available under the accession numbers: synapse20821809.
- As indicated, original code with detailed indications is publicly available at different GitHub repositories.

METHOD DETAILS

Simulation of the cell lineage trees

For both the *C. elegans* and *M. musculus* cell lineages, the mutated/unmutated states of each site were coded in a character state matrix, with each of the possible 30 mutations represented by letters "A-Z" and "a-d", following a random gamma distribution (Box 1). If the target was missing due to an inter-target deletion event or a *dropout* for *M. musculus* (Box 1), the character becomes "-". Once a target is mutated, it can no longer change, either to revert to the unmutated state or to transit to a new state. All mutation schemes can be implemented on any pre-existing tree topology using the software as described here: https://github.com/irepansalvador/Challenge_2

For *C. elegans*, the L3 larval stage cell lineage of the hermaphrodite used for this simulation comes as a *json* format from here: http://wormweb.org/celllineage. Mutation events were implemented as following a Poisson distribution, with a probability of $\mu_t \sim 3x10^{-4}$ mutations per minute. Under the Poisson model, given a mutation rate μ_t per unit time, the probability that a site remains unmutated after t minutes is: $e^{-(\mu_t t)}$. The general implementation of these simulations is similar to the ones described in (Salvador-Martínez et al., 2019). The simulation starts with one cell at minute 0 (fertilization) and ends with 1092 cells at minute 3,065 (L3 stage at 20°C), as cell death was not considered.

For *M. musculus*, the lineage tree and the training sets were constructed using an eSTG program consisting of a set of stochastic tree grammar transition rules that are context-free was used to generate the lineage tree (Spiro and Shapiro, 2016). Cells transition rule probabilities and rates, however, can depend on global parameters such as population size, generation count and elapsed time (Figures 1D and 1E). In addition, each individual cell may have an internal state, which can change during transitions. These rules can be adapted to simulate any cell lineage tree. In the simulation for *M. musculus*, we roughly followed embryogenesis, specialization and proliferation trends in adulthood based on REF, the tree is "sampled" in adulthood at 11 cell types: Cardiac (CD), digestive (DG), lung (LN), neuron (NR), pancreatic (PC), pigment (PG), skeletal (SL), skin (SN), smooth muscle (SM), thyroid (TY), kidney tubule (KT) and in some instances Placenta (PL) (see bottom Figures 1D and S1).

Maintaining the scope and life-like probabilities and rates in the simulation's grammar rules means the simulation quickly scales beyond realistic computing constraints. In order to keep the grammar rules separated from computing constraints we introduced a system of on-the-fly token-based sampling. Each cell internal state was extended with a ''token'' variable that is initiated at the zygote to the number of desired cells in the simulation and inherited by all of its descending cells. In every cell division, the tokens were randomly distributed between the two offspring and when a cell inherited 0 tokens it was excluded from the simulation (Figure S1).

For *M. musculus*, the mutations probability was obtained from a similar distribution as for *C. elegans*, but given the one year time scale, the chance of mutation event was uniformly factored down by 1000 for every character in order for every cell to have a chance of getting at least a scarring event during the one year development. The mutational events were only simulated when calculating cell division events, taking into account the elapsed time from the previous cell division as another uniform factored T for every character.

In both simulated cell lineages the mutation rate was tuned to avoid saturation of the recording array. In the *C. elegans* tree all cells ended up with a different mutational character array whilst in the mouse only ~2.7% of sister cells shared the same character arrays.

Inter-target deletions

In *C. elegans*, if two mutations occur in close targets, (no more than 20 targets apart in the recording array), within a short interval of time (~3 min) during a given cell division, all the targets between them are removed (Box 1).

For *M. musculus*, if two mutations occurred in the same cell, the gamma distribution was then sampled again but this time for the number of sites that might get deleted. Here the distribution was uniformly factored up times 10K to counter the already rare occasion of simultaneous mutations. If the resulting value was greater than 3, a multi-site deletion event was executed on a random stretch of sites. The size of the deletion is exponentially less likely as the number of deleted characters gets larger and as arrays saturate in mutations the chances of multi-site deletion decreases.

Sequencing dropouts

Acquisition dropout distributions (Box 1) were implemented only for the *M. musculus* challenge only. In order to capture the variability of the signal quality in both the individual samples and the different sites we modeled the 'sequencing dropout' of single cell samples by assigning distinct coverage factors for each sample and for each locus. The density of cell coverage factors P = (*pi*: i = 1 to M) is the probability of obtaining a signal in each sample or and the density of site coverage factors Q = (*qj*: j = 1 to N) as the probability of obtaining a signal in each locus. The probability of obtaining a signal in sample i and locus j thus equals *pi.qj.r*. Those are multiplied to get the individual coverage factor of a specific site in a specific cell, finally deriving the acquisition dropout status as a factor of a global coverage parameter *r*.

CHALLENGE BEST PERFORMING METHODS IMPLEMENTATION DETAILS

DCLEAR (Distance based Cell LinEAge Reconstruction)

Weighted Hamming distance (WHD)

DCLEAR (WHD) used the information content to weigh each possible mutated state in the character array 'A-Z' and 'a-c', and the state weight was optimized by Bayesian hyperparameter optimization using the solution trees in the training datasets. The distance matrix between all cells was built using this weighted hamming distance, and the tree then constructed using the Minimum Evolution (FastME) or Neighbor-Joining (NJ) algorithms (Gascuel and Steel, 2006; Lefort et al., 2015).

K-mer replacement distance (KRD)

DCLEAR (KRD) first used the prominence of mutations in the character arrays to estimate the summary statistics that were used for the generation of the tree to be reconstructed such as mutation rate μ, mutation probability for each character in the array, number of targets and number of cells (Figure 4A). These estimated parameters, combined with the pre-defined parameters such as number of cell divisions, were used to simulate multiple lineage trees starting from the unmutated root. To generate trees with a size and mutation distribution similar to the *M.musculus* simulated tree, we generated 1,000 lineage trees with 16 cell divisions or 2^{16} leaves, mutation rate of 0.1, arrays of 200 character, 200 cells and 30 states ('A'-'Z' to 'a'-'c', with outcome probability following a Gamma distribution with shape of 0.1 and rate of 2). Then, different possibilities for the k-mer distances were estimated from the simulated lineage trees and used to compute the distances between input sequences in the character arrays of internal nodes and tips (Figure 4B).

Nodal distance

Nodal distance is defined as the number of steps that separate any two cells in a tree where internal nodes are being considered. For example, in Figure 4B the two red cells have a nodal distance of three. By examining the simulated lineage trees, *DCLEAR* (KRD) estimated the expected *1*-mer distance between characters in the array (including ground state "0" and deletion state "-") in the lineage trees (Figure 4C) and the probability for a given nodal distance of replacing a character in a cell array (Figure 4D). As expected, we observe that smaller nodal distances were associated with rare and unchanged states, so that for example if in a position the arrays of two cells have a "C" then their nodal distance is closer than if they both had an "A" (Figures 4C and 4E). Also, the nodal distance between the ground state or deletion state to any other character is constant (Figure 4C). From the *1*-mer distances and the character replacement probabilities we obtain the nodal distance probability distribution for each character (Figure 4E).

Extending 1-mer to k-mer replacement distance

The approach can be extended to 2,3 or *4*-mers and the posterior probability distributions of nodal distance were estimated by using *an independent model*, that considered the two characters to be independent, and *a conditional model* considering a dependance for the concurrence of mutations (Figures 4F and 4G). Specifically, let us define two *k*-mers $x = x_1,\ldots, x_K$, and $y = y_1,\ldots, y_K$, where x_i and y_i are the single states at position *i* in k-mer *x* and *y*, respectively. The independent model estimates the k-mer replacement distance by 1-mer replacement distance, that is,

$$p(d|\boldsymbol{x},\boldsymbol{y}) \sim \prod_{i=1}^{K} p(x_i,y_i|d)p(d)$$

In comparison, the conditional model estimate the k-mer replacement distance by considering the neighboring states:

$$p(d|\boldsymbol{x},\boldsymbol{y}) \sim \prod_{i=1}^{K} p(x_k,y_k|\boldsymbol{x}_{1:(k-1)},\boldsymbol{y}_{1:(k-1)},d)\cdots p(x_2,y_2|x_1,y_1,d)p(x_1,y_1|d)p(d)$$

In both cases, the expected distances between *k*-mers estimated by the conditional model were more consistent with the observed distribution. This is depicted for the *2*-mers "C-" and "CC" (Figure 4F), and the *4*-mers "BBBB" and "BBBB" (Figure 4G). In the simulation study, we found that k-mer replacement distance (KRD) estimated from longer *k*-mers has better performance (smaller RF distance) than those estimated from shorter *k*-mers or the regular Hamming distance (Figure 4H).

Cell distances

The *DCLEAR* package used KRD estimated by the conditional model to compute the distance between any two sequences of character arrays. Specifically, let us define two sequences $\boldsymbol{u} = \boldsymbol{x}_1,\ldots,x_N$, and, and $\boldsymbol{v} = \boldsymbol{y}_1,\ldots,y_N$, where $\boldsymbol{x}_j$ and $\boldsymbol{y}_j$ are the k-mer at position j in sequence $\boldsymbol{u}$ and $\boldsymbol{u}$, respectively, and N is the number of k-mers (e.g. $200-k+1$ for the *C. elegans* challenge). Assuming that each k-mer is independent, the sequence distance was estimated as:

$$E[d|\boldsymbol{u},\boldsymbol{v}] = \frac{\sum_{j=1}^{N} E\left[d_j|\boldsymbol{x}_j,\boldsymbol{y}_j\right]}{N}$$

The simulated data was used to estimate, distribution of nodal distance $p(d)$, conditional probability of single state replacement $p(\boldsymbol{x},\boldsymbol{y}|d)$, and conditional transition probability of single state replacement $p(\boldsymbol{x}_{i+1},\boldsymbol{y}_{i+1}|\boldsymbol{x}_i,\boldsymbol{y}_i,d)$.

All code is available here: https://github.com/ikwak2/DCLEAR

AMbeRland: Using Machine Learning (ML) for cell lineage reconstruction

We have treated lineage reconstruction as a supervised learning task (see Figure 5). This involved the following steps:

Data preparation

We have used a custom script to extract information on which cells are sister cells from Newick file containing the ground truth cell lineage trees. This produced two lists of cell pairs for each colony: sister cells and non-sister cells. For both lists we picked the corresponding readout of each cell from the text file containing the cell IDs and the values of the character arrays.

Feature engineering

As predictors, we have chosen the following features based on pairwise comparison of recorded arrays: number of recorded units which have not mutated; number of recorded units which have the same mutation; number of recorded units which have single mutation; number of recorded units which have different mutations. The status was set to 1 if two cells were sister cells and to 0 if observed two cells were not sister cells. We included only those pairs with status 0, that have not already been included with status 1. We assumed only binary trees.

Threshold for clustering

The ground truth cell lineage trees were used to evaluate a set of decreasing thresholds corresponding to how any two individual clusters of cells were related at different levels of the lineage tree.

Model training

We used Generalized Boosted Regression (GBR) model to predict the probabilities that two cells are sisters. All calculations were performed in R using package gbm (https://cran.r-project.org/web/packages/gbm/index.html). The following options were used to train Generalized Boosted Regression model: distribution = "bernoulli"; n.trees = 100; interaction.depth = 1; n.minobsinnode = 3; and cv.folds = 5.

Clustering

The clustering starts at the lowest tree level, where all possible pairs of cells are ranked according to the predicted probability that they are sisters. At each consecutive level, pairwise comparison is performed between each lower level cluster by calculating the maximum probability between any elements of two clusters. Pairs of clusters are ordered again according to this probability and are assumed to have the same parent-node if its value is above the estimated threshold for this level. This process was repeated until one or two clusters were left.

All code is available here: https://github.com/rretkute/ml_cell_lineage_reconstruction

Along with aforementioned supervised and unsupervised learning methods, one group also applied Reinforcement learning (RL) to reconstruct the lineage tree by build an *agent* recursively partition the population cells, where the agent's *state* is the set of cells in a particular leaf, the available *actions* are possible splits of those cells into daughter groups, and a neural network was trained to predict the *reward* (how accurately the cells are partitioned) (https://www.synapse.org/#!Synapse:syn21560108/wiki/601091). While this group did not submit the final results, it will be extremely interesting to see how recently developed deep learning techniques can be used for reconstructing lineage trees.

QUANTIFICATION AND STATISTICAL ANALYSIS

Data and software availability

We compiled all the challenge related methods in a wiki: https://github.com/Lineage-Reconstruction-DREAM-Challenge/hub/wiki

Data availability

All challenge datasets and participants submissions are available at: https://www.synapse.org/#!Synapse:syn20821809

Scoring

All scoring scripts are dockerized and available at: https://github.com/Lineage-Reconstruction-DREAM-Challenge/hub/wiki/Scoring

Code for producing the *in silico* datasets

All code for tree and character array mutations simulations is available at:

For the *C. elegans* challenge https://github.com/irepansalvador/Challenge_2

For the *M. musculus* challenge and a guideline for how to adapt the code to other projects: https://github.com/ofirr/clineage-simulation/tree/DREAM

Besides access for all the code for the methods developed in the in vitro challenge being available in Table S2 and the *in silico challenge* in Table S3, all code for the *in silico* challenges best performers of evaluation and reproduction of their methods as applied to the training set in Box 1F is available at: https://github.com/Lineage-Reconstruction-DREAM-Challenge

Code for examples of common tree reconstructions approaches (NJ, TMC, FastTree2) using the challenge dataset can be found here: https://github.com/ofirr/dream_examples