Title

Deep distributed computing to reconstruct extremely large lineage trees

Journal

Nature Biotechnology, 40(4)

ISSN

1087-0156

Authors

Konno, Naoki
Kijima, Yusuke
Watano, Keito
et al.

Publication Date

2022-04-01

DOI

10.1038/s41587-021-01111-2

Peer reviewed

Published in final edited form as:

Nat Biotechnol. 2022 April ; 40(4): 566–575. doi:10.1038/s41587-021-01111-2.

Deep distributed computing to reconstruct extremely large lineage trees

Naoki Konno[1,2,3], **Yusuke Kijima**[1,4,5,10], **Keito Watano**[1,2,3,10], **Soh Ishiguro**[1,5,10], **Keiichiro Ono**[6], **Mamoru Tanaka**[1], **Hideto Mori**[1,7,8], **Nanami Masuyama**[1,5,7,8], **Dexter Pratt**[6], **Trey Ideker**[6,9], **Wataru Iwasaki**[2,3], **Nozomu Yachie**[1,5,7,8]

[1]Synthetic Biology Division, Research Center for Advanced Science and Technology, The University of Tokyo, Tokyo, Japan.

[2]Department of Biological Sciences, Graduate School of Science, The University of Tokyo, Tokyo, Japan.

[3]Department of Integrated Biosciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan.

[4]Department of Aquatic Bioscience, Graduate School of Agricultural and Life Sciences, The University of Tokyo, Tokyo, Japan.

[5]School of Biomedical Engineering, Faculty of Applied Science and Faculty of Medicine, The University of British Columbia, Vancouver, British Columbia, Canada.

[6]Department of Medicine, University of California, San Diego, La Jolla, CA, USA.

[7]Institute for Advanced Biosciences, Keio University, Tsuruoka, Japan.

Correspondence and requests for materials should be addressed to Nozomu Yachie. nozomu.yachie@ubc.ca.

Author contributions

N.K., H.M. and N.Y. conceived the high-level concept of FRACTAL. N.K., W.I. and N.Y. designed the study. N.K. implemented FRACTAL. K.W. and N.K. implemented PRESUME. N.K. led the analyses. N.M. and N.Y. designed the high-content cell lineage recording model. Y.K. and K.W. supported the analyses of the simulated cell lineages. S.I. and M.T. performed the EP-PCR experiments. Y.K. supported the analysis of the EP-PCR experiments. N.K., K.O., D.P. and T.I. performed data visualization using HiView. N.K., Y.K., S.I. and N.Y. wrote the manuscript.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41587-021-01111-2.

Code availability

The source codes and manuals of FRACTAL and PRESUME are available at GitHub (https://github.com/yachielab/FRACTAL and https://github.com/yachielab/PRESUME, respectively). FRACTAL and PRESUME can also be modified and executed on a small scale on web browsers through the Code Ocean, a cloud-based code sharing platform (https://doi.org/10.24433/CO.2433997.v1 and https://doi.org/10.24433/CO.3922773.v1, respectively). Note that the distributed computing mode is disabled in the Code Ocean because the Univa Grid Engine (UGE) is not supported in the Code Ocean. The other codes used in this study are also all available at GitHub (for the list, see Supplementary Table 5).

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s41587-021-01111-2.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41587-021-01111-2.

Peer review information *Nature Biotechnology* thanks the anonymous reviewers for their contribution to the peer review of this work.

[8]Graduate School of Media and Governance, Keio University, Fujisawa, Japan.

[9]Departments of Bioengineering and Computer Science, University of California San Diego, La Jolla, CA, USA.

[10]These authors contributed equally: Yusuke Kijima, Keito Watano, Soh Ishiguro.

Abstract

Phylogeny estimation (the reconstruction of evolutionary trees) has recently been applied to CRISPR-based cell lineage tracing, allowing the developmental history of an individual tissue or organism to be inferred from a large number of mutated sequences in somatic cells. However, current computational methods are not able to construct phylogenetic trees from extremely large numbers of input sequences. Here, we present a deep distributed computing framework to comprehensively trace accurate large lineages (FRACTAL) that substantially enhances the scalability of current lineage estimation software tools. FRACTAL first reconstructs only an upstream lineage of the input sequences and recursively iterates the same produce for its downstream lineages using independent computing nodes. We demonstrate the utility of FRACTAL by reconstructing lineages from >235 million simulated sequences and from >16 million cells from a simulated experiment with a CRISPR system that accumulates mutations during cell proliferation. We also successfully applied FRACTAL to evolutionary tree reconstructions and to an experiment using error-prone PCR (EP-PCR) for large-scale sequence diversification.

There is a limitation in the number of input sequences that can be applied for computational reconstruction of phylogenetic lineage. This limits the capacity to capture the evolutionary trajectories of species and genes to the highest possible extent. Recent progress in the development of computational methods has achieved the reconstruction of a lineage tree for up to approximately 1 million sequences[1]. However, given that the current estimate for the number of unique eukaryotic species on Earth is approximately 8.7 million[2] and the size of their available sequence resources could be multiple orders of magnitude higher than this estimate, it is important to develop more scalable lineage reconstruction methods.

Developmental biology will soon encounter the same computational issues. With the advent of the CRISPR–Cas9 genome-editing technology[3,4], different cell lineage tracing methods have been rapidly developed and applied to analyze the development of multicellular organisms, including mice[5–7], flies[8] and zebrafish[9–12], and cancer proliferation[13,14]. In an ideal CRISPR cell lineage tracing, chromosome-embedded synthetic DNA barcode sequences are continuously and randomly mutated by Cas9 and targeting guide RNAs (gRNAs) and inherited from mother to daughter cells. In this approach, the cell lineage initiated from a fertilized egg can be deduced from the mutation patterns in DNA barcodes of daughter cells identified at the time of observation by high-throughput sequencing and a phylogeny estimation algorithm established in evolutionary biology. This lineage tracing strategy has also been paired with single-cell RNA sequencing (scRNA-seq), allowing for the analysis of cell type differentiation coupled with cell lineage history[10–12]. However, the current methods have not been able to demonstrate high-resolution tracing of large cell lineages, such as mapping of the whole-body cell differentiation trajectories of a mouse

because of the three major technical limitations in (1) continuous recording of lineage information into the high-capacity DNA memory, (2) sampling sensitivity of daughter cells and (3) computational capacity of lineage reconstruction. While technology developments addressing the first two issues have been rapidly progressing with CRISPR[6,13,14] and scRNA-seq technologies[15–17], the third issue remains to be addressed. Considering that mammalian bodies are estimated to be composed of hundreds of millions to trillions of nucleated cells[18], high-resolution lineage tracing methods would need to handle a similar, or multiple orders of magnitude larger, number of mutated sequences for mammalian whole-body cell lineage tracing.

Results

Overview of FRACTAL.

To reconstruct large lineage trees of mutated sequences, we developed a deep distributed computing framework FRACTAL that enhances the scalability of any lineage estimation software and enables incremental top–down reconstruction of a large target lineage, whereby a shallow top-layer lineage is reconstructed first, and the same procedures are recursively iterated in parallel for the next layer of lineages starting from the leaves of its parental lineage by distributed computing (Fig. 1). Generating a hierarchy of distributed computing, FRACTAL resolves large lineage trees in a reasonable computing time.

In FRACTAL, a given number of sequences are first randomly subsampled from an input sequence pool (step 1). Their sample lineage tree is reconstructed with a rooting or provisional rooting sequence by lineage estimation software of choice (step 2) (note that, for cell lineage tracing, while an outgroup or a true ancestral sequence is preferred as the rooting sequence if available, any sequence can be used for unrooted tree estimation). Each of the remaining input sequences is then mapped to its most proximal branch of the sample tree by phylogenetic placement[19,20] (step 3). If all of the input sequences are mapped on the downstream branches of the sample tree to separate them into multiple distinct clades, their upstream lineage is considered to be resolved (step 4), and the sequence group in each downstream clade is recursively subjected to the first process in a distributed computing node (step 5). If any sequence or sequences are mapped on the root branch, which does not allow the grouping of input sequences into clades, the phylogenetic placement is repeated against a new sample tree generated for sequences randomly chosen from a union of the previous subsampled sequences and the ‘problematic’ sequences (step 6). This process generates a new sample tree in a biased manner such that it harbors the previous problematic sequences in its leaves and decreases the probability of acquiring problematic sequences in the subsequent phylogenetic placement step. The retrial process is repeated until the problem is solved but only up to a given threshold number of times and as long as the number of problematic sequences continues to be reduced in every retrial step. When the retrial cycle stops without completely solving the problem, the remaining problematic sequences are discarded, and the other sequence sets are separated into distinct clades and subjected to the first process in separate computing nodes. Accordingly, FRACTAL generates a hierarchy of expanding parallel computing trajectories, where each distributed computing job recursively generates a large set of successive jobs. When the number of input sequences is reduced to

a certain threshold (hereafter called the naive computing threshold), the remaining lineage is directly reconstructed using the designated software, and the operation terminates for this computing trajectory (step 7). For unrooted lineage estimation, the provisional rooting sequence can be removed after completing the entire computation. Thus, FRACTAL enables the efficient reconstruction of large lineages by distributed computing while using limited computing power and memory per node (Supplementary Note 1 and Supplementary Fig. 1).

Lineage reconstruction of 235 million sequences.

To simulate various types of lineages, we also developed a versatile sequence diversification simulator, PRESUME, that generates a large number of mutated sequences (Extended Data Fig. 1 and Methods). In PRESUME, sequences are proliferated by changing their doubling speeds under a stochastic model that determines the balancedness of the producing lineage. Along with proliferation, sequences continuously acquire new mutations according to a user-defined model. We generated a range of lineage trees using PRESUME and demonstrated that FRACTAL with commonly available software tools (RapidNJ[21], RAxML[20] and FastTree[22] for neighbor-joining (NJ), maximum parsimony (MP) and maximum likelihood (ML) methods, respectively) reconstructed them with the accuracies comparable to the original tools (Supplementary Note 2 and Supplementary Figs. 2–5).

Next, to demonstrate the scalability of FRACTAL, we generated an unbalanced lineage of 235,100,752 sequences using PRESUME and reconstructed the lineage using FRACTALized RapidNJ using 300 computing nodes, each with 128 gigabytes (GB) of memory. This process required a median of four recursive FRACTAL iterations that produced a total of 39,840 jobs (Fig. 2a–d) and a total runtime of 31.6 h (Fig. 2e). The reconstructed lineage involved nearly 100% of the input sequences (235,100,199). Because there is no computing method to directly measure the accuracy of the whole lineage reconstruction by comparing the topological agreement of the reconstructed and simulated ground truth trees at this scale, we randomly sampled arbitrary numbers of sequences included in the reconstructed tree and measured topological agreement of their subgraph trees with the simulated tree using normalized Robinson– Foulds distance (NRFD)[23] (Methods; Fig. 2f). In this analysis, a steady increase in topological agreement was evident from a relatively large sample size of sequences from approximately 10^4, which ended with an agreement score of over 99.8% for the maximum tested size of 10^6 sequences. This result suggests that the overall accuracy of this reconstructed lineage was over 99.8%. Direct lineage estimations were also performed for the same sets of subsampled sequences using the original tool (RapidNJ). Interestingly, the topological agreements of the subgraph trees reconstructed by the original tool with the simulated ground truth tree were markedly lower than those in the entire tree estimated using FRACTAL. Although the agreement score outputted by the original tool constantly increased with the input sequence size, it reached only 89.4% for the maximum capable number of sequences (100,000) with the same memory limitation per node, whereas the corresponding subgraph in the entire tree reconstructed by FRACTAL displayed an agreement score of 99.6%. This supported the important suggestion that the lineage estimation accuracy for a set of target sequences increases when other sequences produced in the same sequence diversification process are fully taken into account.

The recursive FRACTAL iteration involves three major time-consuming steps: (1) sequence subsampling from the entire input sequence, (2) phylogenetic placement and (3) sorting of sequences mapped on each of the sample lineage clades for successive job cycles. While the present implementation of FRACTAL uses another layer of distributed computing with available computing nodes only for the phylogenetic placement step and a part of the subclade grouping step after aggregating the phylogenetic placement result using a single computing node (Fig. 2g), the computing loads of the remaining steps can also theoretically be distributed. To estimate the scalability of FRACTAL with this best possible implementation, we developed a simple FRACTAL runtime simulator, in which we modeled the input sequence size-dependent runtimes of FRACTAL iterations and runtime-dependent occupancies and releases of computing nodes in a distributed computing environment (Extended Data Fig. 2). The sequence size-dependent runtime function was fit to runtime log data for which all three steps in each FRACTAL iteration were processed by a single node in the lineage reconstruction of 235 million sequences. We then constructed a FRACTAL runtime simulator in which no secondary distributed computing was used in any of the three steps (model A). Another runtime simulator (model B) was constructed in which all three steps were processed by distributed computing, assuming that the number of available free computing nodes can linearly reduce the runtime of each FRACTAL iteration. Model B suggested that the runtime of FRACTAL with ideal implementation would be proportional to the number of input sequences and inversely proportional to the number of computing nodes (Fig. 2h). Furthermore, models A and B predicted total runtimes of 449.6 and 12.2 h, respectively, to reconstruct a lineage of 235 million sequences with 300 computing nodes. These predicted runtimes were consistent with the runtime of the present implementation (31.6 h), which was intermediate between the two models, suggesting room for improvement.

Reconstruction of evolutionary and pseudo-evolutionary trees.

The lineage tree for 13,897 16S rRNA sequences obtained from the SILVA database[24] (Fig. 3a) was also estimated using the original phylogeny estimation tools and those FRACTALized with different subsampling and naive computing threshold parameters. Compared with the gold standard lineage of SILVA 16S rRNA sequences, the performance of FRACTAL was similar to that of the original tools (Fig. 3b). However, this experiment did not satisfy the fundamental question of how accurately FRACTAL can reconstruct evolutionary lineages because the SILVA 16S rRNA lineage was also generated by a prediction, and no ground truth lineage for any evolutionary process was reported. Furthermore, there has been no large biologically relevant lineage available that involves more than 1 million sequences and enables examination of the scalability of FRACTAL. Thus, we fit the PRESUME parameters to the reported evolutionary tree datasets to obtain a large pseudo-evolutionary tree. We first calibrated the parameter of PRESUME that determines its fluctuation level in the sequence proliferation speeds (topological lineage balancedness) so that the distribution of branching balance indices (BBIs; Methods) of the simulated lineage was similar to or higher than those of the SILVA 16S rRNA lineage tree and the Genome Taxonomy Database (GTDB) reference trees[25] (Fig. 3c,d). The uniformity parameter for mutational positions and nucleotide substitution rate parameters of the GTR-Gamma model were then separately estimated using the SILVA 23S rRNA dataset, which

consists of longer sequences than 16S rRNAs. At the same time, using the optimized topological balancedness parameter, we generated a tree of 1,019,509 leaves such that its relative branch length distribution was the same as that of the SILVA 23S rRNA lineage. Finally, the overall mutational level parameter of the GTR-Gamma model was fitted to capture the diversity of the sequences in the SILVA 23S rRNA sequences (Fig. 3e,f).

For the pseudo-evolutionary lineage of 1,019,509 sequences (Fig. 3g), we examined the accuracies of the different phylogeny estimation tools and those FRACTALized with single and 100 computing nodes to reconstruct various sizes of its subclades (Fig. 3h). Overall, any of the FRACTALized lineage estimations could reconstruct the whole lineage. Their accuracies were similar to or slightly higher than the original tools for any size lineage reconstruction. To examine the robustness of FRACTAL in reconstructing large pseudo-evolutionary trees, three more pseudo-evolutionary lineages of 1,019,509 sequences were simulated without changing the lineage tree structure. Independent lineage estimations were performed five times for each of the four large sequence datasets using FRACTAL with 100 computing nodes. These experiments demonstrated that FRACTAL robustly reconstructed the large pseudo-evolutionary trees with accuracies similar to or higher than those of the original tools for their maximum capable lineage sizes (Fig. 3i). We also demonstrated that FRACTALized phylogeny estimations were also as robust as the commonly used software tools against noise sequence contamination. For the SILVA 16S rRNA sequences mixed with 0% to 20% random sequences, the reconstruction accuracies of their gold standard lineage remained the same for RapidNJ, RAxML and FastTree and those FRACTALized (Extended Data Fig. 3)

FRACTAL for high-resolution tracing of large cell lineages.

Because the idea of CRISPR cell lineage tracing has been beautifully demonstrated[9,26], the area has been rapidly developing. However, no method has yet achieved high-resolution tracing of large cell lineages, such as the entire cell lineage of animal body development. Several methods have used multiple evolving barcodes integrated into each cell, in which multiple barcodes in distal chromosomal regions independently acquire mutations[5,6,13,14,26,27]. In such a system, the barcodes are tagged with a unique static identifier (or locus ID) and transcribed under polymerase II promoters, allowing for combinatorial mutation patterns in individual cells to be read by scRNA-seq or spatial genomic approaches. While the current systems use either a few copies of a CRISPR targeting array or many recording units of a limited lineage recording capacity, one plausible way to increase the memory capacity for high-resolution cell lineage tracing is a combination of these strategies, where each cell has dozens to hundreds of transcribing barcode arrays of multiple CRISPR targeting units (Fig. 4a). If realized, this approach would also allow a quantitative self-evaluation of reconstructed lineages without a ground truth lineage (Fig. 4b). DNA barcode arrays can be split into two distinct sets. Therefore, the agreement between cell lineages orthogonally reconstructed from independent datasets would represent the accuracy of the reconstructed lineages. Furthermore, in current technologies, the use of Cas9 to induce continuous mutations also has major drawbacks. Cas9-based induction of DNA double-stranded breaks (DSBs) to introduce mutations is toxic to cells[28], and DSBs often induce deletions after non-homologous end-joining repair.

Furthermore, multiple DSBs at a DNA barcode array, if they occur simultaneously, would pop out the entire memory region sandwiched by the cut sites. These deletions result in the termination of the mutation process and mask lineage information recorded before such events. One strategy to mitigate this effect would be the use of CRISPR base editors that induce nucleotide substitutions with minimal DSB activities[29]. Several studies have recently established dual-function CRISPR base editors with C→T and A→G activities, which can greatly diversify sequences in a narrow window of the gRNA target sequence[30–32].

Foreseeing these next-generation strategies involving the high-resolution lineage tracing of a large number of cells, we examined whether FRACTAL can practically handle such a game. Using the genome editing outcome dataset of scGESTALT generated for the tracing of zebrafish brain cell lineages[10] and the base editing spectrum data of dual-function base editor Target-ACEmax that we developed previously[32], PRESUME was modeled to simulate a lineage of over 16 million cells each with 100 copies of a barcode array that was continuously mutated. Each barcode array was designed to have nine gRNA target sites and a unique locus identifier. Because currently published CRISPR cell lineage tracing data are dominated mainly by multifurcation branches and do not enable quantitative estimation of the unbalancedness of a typical vertebrate cell lineage (Extended Data Fig. 4a), we obtained an embryonic cell lineage of *C. elegans*[33] (Fig. 4c), examined its branching balancedness and used a topological parameter for PRESUME that ensured a more unbalanced sequence diversification process than *C. elegans* embryogenesis (Fig. 4d).

To model substitutions by Target-ACEmax, the average C→T and A→G base editing spectra across different positions of a gRNA-targeting unit were obtained from the previous dataset[32] with their scaling factors to represent base editing efficiencies of different target sequences (Fig. 4e). Different base editing frequencies across the array of nine gRNA targets were then modeled according to the base editing scaling factor distribution. Finally, we used a constant secondary base editing scaling factor to be multiplied by these base editing frequencies at different sequence positions to represent those per cell generation. A minimized level of insertions and deletions (indels) has been reported to be induced by base editing. To model the unique mutation patterns in the barcode array with overestimated frequencies of base editing-induced indels, we obtained frequencies of observed insertion positions and deletion center positions across the array of nine gRNA targets and their observed size distributions in the scGESTALT dataset[10] as their relative event probabilities and size distributions of independent events (Fig. 4f).

Using the mutation model, we first calibrated the indel probabilities per generation so the average total insertion and deletion sizes of each array conferred by a simulation of producing 4,000 cells would be similar to those observed in the scGESTALT dataset (Extended Data Fig. 4b,c). Assuming that the average total generation time from the root to leaves of a given tree was double that of a tree whose number of leaves is square rooted, half of these indel probabilities were allocated to simulate the lineage of 16 million ($4{,}000^2$) cells such that each producing cell would acquire a similar degree of indels. To determine the optimal secondary base editing scaling factor for the lineage of 16 million cells, we also simulated the production of 4,000 cells and found that lineage tracing by RapidNJ performed the best for a secondary base editing scaling factor of approximately

0.5 (Extended Data Fig. 4d–f). Accordingly, we simulated lineages of 16 million cells using a secondary base editing scaling factor of 0.25 (Fig. 4g) and an order of magnitude lower scaling factor of 0.025 (Fig. 4h). These base editing frequencies per generation were considered reasonable because of the secondary base editing factor of 1.0, which is equivalent to the training base editing dataset obtained after three generations of HEK293Ta cells[32], and because the base editing efficiency could be easily attenuated by changing promoters or mutating the base editor or gRNA.

Using FRACTALized RapidNJ with 100 computing nodes, the simulated cell lineages of 16 million cells were reconstructed from different numbers of barcode arrays (Fig. 4i). For both base editing frequency levels, the accuracy of lineage reconstruction measured by comparing to the ground truth lineage of simulation increased with the number of barcode arrays used, as expected (Fig. 4j). For the secondary base editing factor of 0.25, the lineage reconstruction accuracy exceeded 99% when 40 barcode arrays were used. The maximum accuracy of 99.5% was observed with the total runtime of 16.5 h when all 100 barcode arrays were used. Even for a scaling factor of 0.025, the accuracy exceeded 99% with the total runtime of 16.7 h when all 100 barcode arrays were used, showing that FRACTAL was amenable for high-resolution tracing of large cell lineages with many barcode arrays (Fig. 4j). Furthermore, the agreement of orthogonal cell lineage trees reconstructed from independent sets of barcode arrays increased with the number of barcode arrays used to reconstruct each tree (Fig. 4k). The agreement levels of orthogonally reconstructed trees were linearly correlated with the accuracies of lineages reconstructed using the same number of barcode arrays (Fig. 4l). This result suggests that the orthogonal lineage reconstruction strategy would enable a quantitative estimation of lineage accuracy when a ground truth lineage is unavailable, which is expected in developmental cell lineage tracing. We also plugged Cassiopeia[27], a recently developed cell lineage tracing software, into FRACTAL and demonstrated that both Cassiopeia and FRACTALized Cassiopeia underperformed FRACTALized RapidNJ and FRACTALized RAxML to reconstruct cell lineages with many barcode arrays (Fig. 4m, Supplementary Note 3 and Supplementary Figs. 6–11).

Tracing large sequence diversification processes in EP-PCR.

To examine whether FRACTAL can also trace experimentally produced sequence diversification processes, we performed two-step EP-PCR experiments and generated two mutated sequence datasets of a *CTNNB1*-encoding region (198 base pairs (bp)) and an arbitrarily selected sequence BET002 (110 bp) in the human genome, each with 96 designed lineage bottlenecks. The generation of each dataset was initiated by subjecting an input sequence clone to an EP-PCR (Fig. 5a). Following bacterial cloning of the PCR product and isolation of 96 colonies, the plasmids were purified and subjected to secondary EP-PCR. The same 96 plasmid samples were subjected to high-fidelity PCR (HF-PCR). The PCR products were uniquely indexed and pooled for sequencing. Sequencing reads were demultiplexed, and unique reads were identified for each sample. The sequencing reads of HF-PCR were used to identify the parental first-generation sequences in each well subjected to secondary EP-PCR. We found multiple first-generation sequences in some wells, suggesting imperfect clone isolation. Most of the second-generation sequences showed the best match to the parental first-generation sequences identified for the corresponding wells,

demonstrating that the experiment generated expected datasets (Fig. 5b and Extended Data Fig. 5).

We obtained 3,077,052 and 3,146,330 unique second-generation sequences for *CTNNB1* and BET002, respectively. In this study, we also invented an incremental multiple sequence alignment (MSA) method for FRACTAL, which does not require a preliminary MSA of the entire input sequences (Extended Data Fig. 6 and Supplementary Note 4) and reconstructed the *CTNNB1* and BET002 lineages using this method. The total runtimes for *CTNNB1* and BET002 were 41.9 h and 32.6 h, respectively. The reconstructed lineages covered most of the input sequences (3,020,109 and 3,119,911 sequences for *CTNNB1* and BET002, respectively) and showed significant clustering of sequences observed from the same wells (Extended Data Fig. 7). Under the assumption that second-generation sequences that best matched the parental sequences of different or multiple source wells were derived from sample cross-contamination or insufficient edit distance conferred by EP-PCR, these sequences were removed from the reconstructed lineages, leaving 1,983,716 and 1,483,691 sequences for *CTNNB1* and BET002, respectively (Extended Data Fig. 6). This improved clustering of same-well sequences for both datasets (Fig. 5c–f and Extended Data Fig. 5). Especially for *CTNNB1*, the downstream clades consisting of 1,000 to 15,000 sequences were dominated mainly by sequences from single wells (Fig. 5e). Furthermore, the distances of sequence pairs in the reconstructed lineages were shorter for sequences identified for the same wells than for sequences in different wells (Fig. 5g and Extended Data Fig. 5). Consistent with the fact that *CTNNB1* sequences accumulated more mutations than BET002 (Extended Data Fig. 5), these results also demonstrated that the reconstructed lineage of *CTNNB1* had a higher agreement with the experimental sample manipulation process than BET002. FRACTAL could also reasonably reconstruct lineages of at least up to approximately 3 million experimentally produced sequences. This scale was not possible using the original lineage estimation software tools.

Discussion

In evolutionary biology, phylogenetic trajectories of species have been estimated by lineage reconstructions for independent taxonomic categories at different ranks. The definitions and relationships of these ranks have been empirically given by lineage reconstruction of representative species or by morphological analyses[34]. However, recent metagenomic sequencing efforts have revealed the existence of the large candidate phyla radiation with an unidentified clade size. Its members are largely uncultivable, resulting in difficulties in morphological or other phenotypic analyses[35]. Considering that there are also many other potential underexposed clades, it is no longer unusual that massive amounts of sequence data for novel species are acquired before identifying their taxonomic positions. This growing trend requires large-scale, unsupervised phylogenetic estimation using only input sequence information. The deep distributed computing strategy FRACTAL would have great potential to reveal the vast evolutionary landscape of species that have emerged on Earth.

The success of FRACTAL shows that random sampling of sequences is adequate for precise estimation of only an upper layer of a target lineage. At the same time, when different fractions of sequences were randomly sampled from the simulated dataset of

235 million sequences for independent small-lineage reconstructions, the accuracies of the estimated lineages largely underperformed compared to the corresponding lineage subgraphs in the whole reconstructed lineage, showing that the sampling strategy was inferior in reconstructing the downstream lineage. This phenomenon has already been seen at a much smaller scale in evolutionary biology; a lineage estimation of species from a taxonomic group is more successful when species are densely sampled from the taxonomic group at their full extent[36]. We found that this is still the case even for an unbiased sampling of tens of thousands of sequences from hundreds of millions of sequences. This indicates the need to use sequences from as many species as possible, even for large-scale phylogeny estimation in evolutionary biology, where FRACTAL would significantly contribute. This also has a striking impact on developmental cell lineage tracing and raises the need for whole-body, high-resolution cell lineage tracing, even when a limited number of cells are being investigated.

The genetic circuits for the high-resolution lineage recording would be realized in a way proposed in the present study. The barcode array of multiple gRNA-targeting units[9,10], the use of multiple recording units per cell for capture by scRNA-seq[6,7,10] or spatial genomics[26,37], the introduction of dozens to hundreds of recording units per cell[5] and the use of base editors[29] have all been independently demonstrated. The sampling sensitivity of daughter cells has been a conceptual bottleneck for the high-resolution tracing of large cell lineages. However, the speed of scRNA-seq technologies has been increasing faster than predicted by Moore's law in the last 6 years since the advent of droplet-based scRNA-seq methods[38,39], which enabled the analysis of multiple thousands of cells. The current scalability of split pool barcoding-based scRNA-seq is now on the order of multiple millions of cells[16,17]. It is not too optimistic to predict that technologies that capture multiple orders of magnitude higher numbers of cells will appear in the next 5 years. Thus, the deep distributed computing presented in this study will break the last wall hindering high-content lineage analysis. Furthermore, unlike in the field of evolutionary biology that can be satisfied with a one-time reconstruction of high-resolution lineages of target organisms, FRACTAL will be of greater utility when large-scale lineage estimations are repeatedly required to explore developmental processes of multiple individuals from the same species and those of different species. The reconstruction of individual lineages would lead to the identification of deterministic and stochastic cell differentiation and body formation processes.

In summary, we developed a new deep distributed computing method, FRACTAL, for the scalable reconstruction of extremely large lineages and verified its validity by successfully reconstructing various types of large lineage trees that the commonly used software tools could not reconstruct. Although many lineage estimation tools have been developed in the long history of evolutionary biology, all of the major tools use algorithms to reconstruct an entire tree all at once with the need to continuously monitor the computing status of all of the input sequences, preventing their computing processes from being subdivided into isolated tasks. Instead, FRACTAL abandons the reconstruction of the target lineage all at once and incrementally pursues independent estimations of upstream lineages that can be precisely reconstructed from sparsely subsampled sequences. The computing costs of FRACTAL, even with a single node, were successfully reduced from those of the original

tools in many cases, suggesting that the current lineage estimation algorithms are overkill for many practical tasks. The strategic shift in FRACTAL has dramatically reduced the overall computing cost, allowed the use of distributed computing and essentially enhanced the lineage estimation capability. This concept will be fundamental for large-scale lineage reconstruction using distributed computing and will benefit both evolutionary biology and developmental biology, whose data resources have been growing rapidly.

Methods

Datasets.

The SILVA 16S and 23S rRNA trees and their MSA results were obtained from the SILVA database[24] (LTPs132_SSU and LTPs123_LSU, downloaded on 13 December and 18 December 2020, respectively). Sequence alignment gaps of LTPs132_SSU and LTPs123_LSU were trimmed using trimAl 1.4rev15 (ref. [42]) with '-gappyout'. The GTDB reference prokaryotic lineages were obtained from the GTDB[25] (release 95; downloaded 18 December 2020). As some software did not accept the RNA sequence representation, uracils in the sequences were converted to thymines. The embryonic cell lineage of *C. elegans* was generated from time-lapse three-dimensional measurement data of embryonic cell divisions obtained from a previous study (http://www.digital-development.org/; dataset name: ZD_RW10348_WT_20110126_2_s1_emb2)[33], and the data were converted into a Newick file such that each branch length represents the duration of its corresponding cell imaged with a resolution of 75 s (Supplementary Data 1). The base editing outcome data of Target-ACEmax at 47 gRNA target sites in the human genome (HEK293Ta cells 3 d after transfection) were obtained from our previous study[32]. The scGESTALT barcode sequencing results for cell lineage tracing of zebrafish brain development were obtained from the National Institute of Health Sequence Read Archive (SRR6176748, SRR6176749 and SRR6176750 for ZF1, ZF2 and ZF3, respectively). The benchmarking datasets for evaluating cell lineage tracing performance prepared for Cassiopeia were downloaded from the Zendo repository (https://zenodo.org/record/3706351).

FRACTAL.

Execution.—Lineage estimation using FRACTAL is executed with the information of input sequences, a rooting sequence, a lineage estimation software of choice, subsampling size k, naive computing threshold t, phylogenetic placement method of choice, subgraph size for phylogenetic placement z, the maximum number of sample tree reconstruction retrials x and the number of computing nodes d. The input sequences are provided in a FASTA-formatted file. The user can provide an MSA result of the input sequences in the FASTA file by representing gaps by '-' or choosing an option to incrementally perform MSA along with the FRACTAL process. Any lineage estimation software tool can be plugged into the sample tree and terminal tree reconstructions using a user-prepared script. In the present implementation of FRACTAL, RapidNJ 2.3.2 (ref. [21]), RAxML 8.2.12 (SSE3, HPC, Pthreads version)[20] and FastTree 2.1.10 (SSE3, OpenMP version)[22] were implemented as default lineage estimation tools for NJ, MP and ML, respectively (note that while RAxML is commonly used for ML-based phylogeny estimation from a starting tree reconstructed by MP, FRACTAL uses this software to reconstruct MP trees by

inactivating the likelihood optimization). For phylogenetic placement, RAxML 8.2.12 (MP) or EPA-ng 0.3.5 (ML)[19] was selected for the present implementation of FRACTAL. The GTRCAT model of EPA-ng was used for phylogenetic placement throughout the present study. The user-provided rooting sequence is commonly used for all sample and terminal tree reconstruction processes.

FRACTAL iteration.—In each FRACTAL iteration, if the number of input sequences n is equal to or less than t, the lineage of the entire input sequence is directly estimated by the software tool of choice, and the operation for that computing trajectory terminates. Otherwise, k sequences are randomly subsampled from the input sequence pool. Redundant sequences are removed from the subsampled sequence pool, and unique sequences are used to reconstruct a sample tree by the lineage reconstruction tool plugged in by the user. All of the input sequences are then independently mapped to their most proximal branches of the sample tree using the phylogenetic placement method of choice. When $z < k$, a subtree of the sample tree for randomly selected z sequences is used for phylogenetic placement to reduce memory cost. If multiple branches were found to be equally parsimonious for an input sequence (for MP-based phylogenetic placement only), a random branch is selected.

If all of the input sequences are placed on downstream branches of the sample tree such that they are separated into multiple distinct clades, their upstream lineage is considered to be resolved and fixed. The sequence groups in different downstream clades are then recursively subjected to the next FRACTAL iterations using different computing nodes in parallel. If any sequences are placed on the branch connecting to the rooting sequence, thereby preventing the grouping of input sequences into clades, the sample tree reconstruction and phylogenetic placement processes are repeated by subsampling k sequences from a union of the previously subsampled sequences and the ‘problematic’ sequences. This process is repeated until the problem is solved, but only up to x times as long as the number of problematic sequences continues to be reduced in every retrial step. When the retrial cycle stops, problematic sequences on the rooting branch, if any, are discarded, and the other sequences are grouped into distinct clades for the next FRACTAL iterations.

The present implementation of FRACTAL uses another layer of distributed computing to accelerate the computing speed. The initial input sequences are first split into multiple FASTA files using SeqKit 0.12.1 (ref. [43]) before the first FRACTAL cycle. The number of split files becomes equal to the number of computing nodes d assigned for the FRACTAL run. Followed by random subsampling of sequences from randomly selected files and sample tree reconstruction, the input sequence files are independently subjected to available computing nodes for phylogenetic placement. The placement results from the different computing nodes are aggregated to determine an upstream lineage using a single computing node. Grouping of sequences into downstream clades for the next FRACTAL iteration is also performed independently for the input sequence files under distributed computing. For each of the determined clades, multiple sequence files are generated from the distributed computing nodes and directly subjected to successive iterations.

Lineage reconstruction using both substitutions and indels.—While most of the practical phylogeny estimation tools can consider only substitutions, FRACTAL allows any

software of choice to use both substitutions and indels for scalable lineage reconstruction. In this option, similar to a previously described method[9,10], mutation patterns in the input sequences are converted into another sequence representation using four nucleotide sequences in the sample tree reconstruction and phylogenetic placement of each cycle as follows. All of the mutation patterns in sequences subsampled at each FRACTAL iteration are first encoded by binary letter patterns of unique positions to convert each mutated source sequence into a binary sequence of a fixed length, each of which represents the presence and absence of a specific mutation observed in the original subsampled pool by 1 and 0, respectively. The binary sequence of 1/0 is then further converted into a nucleotide sequence of thymine/cytosine or adenine/guanine at each position. The converted sequences are used to construct a sample tree using the given software of choice. Phylogenetic placement is achieved by converting the remaining sequences using the same encoding rule, ignoring the unique mutation patterns observed outside the original subsampled pool. The encoding rule is updated for every sample tree reconstruction, economically taking into account all mutations in the original input sequences through the lineage reconstruction process. Terminal tree reconstruction is performed using the same sequence conversion process. Because RAxML only accepts input sequences containing all four nucleotide characters, we used the thymine/cytosine and adenine/guanine codes in the upstream and downstream halves of the converted sequences, respectively.

Regulations implemented for effective distributed computing.—To prevent unnecessary increases in overhead costs of waiting time and process management in distributed computing, the present version of FRACTAL includes several simple regulations. In the phylogenetic placement of each FRACTAL iteration, the independent tasks of placing n sequences to a sample tree are also performed using distributed computing. To save available computing nodes for other processes running in parallel, this process is subdivided into $\max(\lfloor d \times n/n_0 \rfloor - 1, 1)$ groups for different computing nodes, where n_0 is the number of input sequences subjected to the initial FRACTAL cycle. In the subclade grouping process after phylogenetic placement, an upper tree is defined for branch depths of up to $\lfloor 1 + \log_2(n/t) \rfloor$ to prevent small sequence groups that can be easily handled by the terminal lineage reconstruction from being unnecessarily split into many smaller groups for the next FRACTAL iterations. Furthermore, when $n \leq n_0/d$, the following FRACTAL iterations are performed in the same computing node without distributing the tasks to the other computing nodes.

The FRACTAL parameters used in all experiments in the present study are listed in Supplementary Table 1.

Runtime simulation of FRACTAL.

The runtime simulator of FRACTAL in a distributed computing environment was modeled based on the input size-dependent runtime of each FRACTAL iteration cycle and a runtime-dependent occupation and release of computing nodes. In this simulation, assuming a high-performance computing environment with a fixed number of distributed computing nodes d, each FRACTAL iteration cycle of input sequence size n starts if there is an available computing node; otherwise, it is stalled until a newly available node is released

from one of the ongoing jobs. Each FRACTAL job process occupies one computing node for a runtime $f(n)$. The incremental FRACTAL job propagation process was modeled with a simple assumption where every FRACTAL iteration produces two new child jobs each with an input sequence size of $\lfloor n/2 \rfloor$. When n is equal to or less than the naive computing threshold t, the job terminates after occupying one node for a runtime $f(n)$, assuming the terminal lineage reconstruction process.

Each FRACTAL iteration involves three major steps that require high computing loads: (1) sequence subsampling from the entire input sequences, (2) phylogenetic placement and (3) subclade grouping of input sequences for successive job cycles. While these computing loads can, in theory, be distributed into multiple computing nodes, FRACTAL implemented in the present study uses another layer of distributed computing only for the phylogenetic placement step and a part of the subclade grouping step after aggregating the phylogenetic placement result using a single computing node. We first fitted the $f(n)$ function for the model in which none of the three steps is processed by distributed computing (model A) using runtime log data of independent FRACTAL iteration cycles that were processed without the secondary distributed computing in the lineage reconstruction of the approximately 235 million sequences. Next, we modeled the best implementation of FRACTAL, where the collective computing load $f(n)$ from the three major steps is perfectly distributed using the available computing nodes (model B) as follows:

$$\frac{f(n)}{\max\left(1, d \times \frac{n}{n_0}\right)}$$

Here, the denominator represents the number of available computing nodes, assuming that the smaller the input sequences for a FRACTAL iteration cycle the higher the chance of computing nodes being occupied by the other job processes at the same period on a linear scale.

PRESUME.

Execution.—PRESUME (Supplementary Fig. 2) enables the simulation of various sequence diversification processes based on user-defined models. It is executed with the input information of a root sequence, a target number of sequences to be generated n as well as a parameter σ that determines the branch unbalancedness of the generating tree and parameters that determine how to introduce mutations in sequences. A template tree with defined branch lengths (or generation times) can also be given, such that the mutational process is simulated along with the provided tree.

Simulation of tree topology.—Unless a tree of defined topology is provided by the user, PRESUME first simulates a template tree topology for the proliferation of propagating units (PUs) (that is, sequences or cells as units to harbor multiple sequences) using a single parameter σ. Following a linear time T progression, PRESUME progressively proliferates PUs, in which each PU duplicates when its generation time d has passed since its birth. The generation time d is given to each new PU, so its reciprocal (doubling speed) follows a gamma probability distribution, wherein the mean and variance are 1 and σ^2, respectively.

$$1/d \sim \Gamma\left(1/\sigma^2, \sigma^2\right).$$

When the distributed computing mode is disabled, the simulation stops at time $T = t$ when the number of generated PUs satisfies $\geq n$ (note that some PUs can be generated at the same time). When the distributed computing mode is enabled, PRESUME first simulates a propagation process using a single computing node $T = t'$ when the number of generated PUs satisfies $\geq \sqrt{n}$ the propagation until process starting from each of these PUs is then operated independently by distributed computing until $T = 2t'$ so that the total number of generated PUs is $\sim n$.

Nucleotide substitution.—When a template tree is provided by the simulation or by the user, the mutational processes of sequences are simulated along with it. In the present implementation of PRESUME, substitutions are simulated either with time-dependent probability functions assigned for different sequence positions using the GTR-Gamma model or time-independent probabilities per branch (or generation) defined for independent sequence positions by the user.

When the GTR-Gamma model is chosen, a position-specific relative substitution parameter γ_i for every different position i is first determined by a gamma distribution with user-defined parameters μ and α as follows:

$$\gamma_i \sim \Gamma(\alpha, /\alpha)$$

Let $P_i(\Delta t)$ be a 4 × 4 matrix, where $P_i(\Delta t)_{x \to y}$ is the transition probability from a certain source nucleotide x to a destination nucleotide $y (x, y \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\})$ within the time interval Δt. $P_i(\Delta t)$ is then defined using γ_i and the substitution rate matrix Q as follows:

$$P_i(\Delta t) = e^{\gamma_i \Delta t Q}$$

where Q is given by

$$\mathrm{Q} = \begin{pmatrix} - & a_{\mathrm{A}\to\mathrm{C}} & a_{\mathrm{A}\to\mathrm{G}} & a_{\mathrm{A}\to\mathrm{T}} \\ a_{\mathrm{C}\to\mathrm{A}} & - & a_{\mathrm{C}\to\mathrm{G}} & a_{\mathrm{C}\to\mathrm{T}} \\ a_{\mathrm{G}\to\mathrm{A}} & a_{\mathrm{G}\to\mathrm{C}} & - & a_{\mathrm{G}\to\mathrm{T}} \\ a_{\mathrm{T}\to\mathrm{A}} & a_{\mathrm{T}\to\mathrm{C}} & a_{\mathrm{T}\to\mathrm{G}} & - \end{pmatrix} \begin{pmatrix} \pi_{\mathrm{A}} & 0 & 0 & 0 \\ 0 & \pi_{\mathrm{C}} & 0 & 0 \\ 0 & 0 & \pi_{\mathrm{G}} & 0 \\ 0 & 0 & 0 & \pi_{\mathrm{T}} \end{pmatrix}$$

The sum of the diagonal values of the right-side matrix of Q must be 1 ($\pi_{\mathrm{A}} + \pi_{\mathrm{C}} + \pi_{\mathrm{G}} + \pi_{\mathrm{T}} = 1$), and the left-side matrix needs to be symmetrical (that is, the same values are assigned to the symmetrical nucleotide transition patterns), wherein the diagonal missing values are given to satisfy the condition that every row sum of Q becomes 0. Here, $a_{x \to y}$ and π_x are user-defined parameters.

Alternatively, the user can define time-independent substitutions per branch (or generation) using the substitution probability matrix $\Phi_{\mathrm{sub},i}$ for every sequence position whose original position in the root sequence is i, as follows:

$$\Phi_{\mathrm{sub},i} = \begin{pmatrix} - & \Phi_{i,\mathrm{A}\to\mathrm{C}} & \Phi_{i,\mathrm{A}\to\mathrm{G}} & \Phi_{i,\mathrm{A}\to\mathrm{T}} \\ \Phi_{i,\mathrm{C}\to\mathrm{A}} & - & \Phi_{i,\mathrm{C}\to\mathrm{G}} & \Phi_{i,\mathrm{C}\to\mathrm{T}} \\ \Phi_{i,\mathrm{G}\to\mathrm{A}} & \Phi_{i,\mathrm{G}\to\mathrm{C}} & - & \Phi_{i,\mathrm{G}\to\mathrm{T}} \\ \Phi_{i,\mathrm{T}\to\mathrm{A}} & \Phi_{i,\mathrm{T}\to\mathrm{C}} & \Phi_{i,\mathrm{T}\to\mathrm{G}} & - \end{pmatrix}$$

Here, all elements in the matrix are non-negative, and the diagonal missing values are given to satisfy the condition that every row sum becomes 1.

Indels.—The time-independent indel simulation is as follows. Sequences are propagated along the template PU tree and progressively accumulate indels at each generation. Let i' be the nucleotide position of a sequence at a given branch of the PU tree, where its corresponding position in the root sequence is i. Insertion events in each generation (branch) are modeled by insertion probability parameters $\Phi_{\mathrm{ins},i}$, representing the chance of having an insertion event at the $3'$ side of nucleotide position i' and a position-independent size probability distribution $\psi_{\mathrm{ins}}(L)$. The sequence of the insert length L at every event is randomly generated with equal probability for each nucleotide at every position. Deletion events are modeled by deletion probability parameters $\Phi_{\mathrm{del},i}$ and a position-independent size probability distribution $\psi_{\mathrm{del}}(L)$, where a deletion of size L centering at a nucleotide position i' spanning from position $(i' - \lfloor L/2 \rfloor)$ through $(i' - \lfloor L/2 \rfloor + L - 1)$ occurs with probability $\Phi_{\mathrm{del},i}$, where L is stochastically determined according to $\psi_{\mathrm{del}}(L)$. Note that $\Phi_{\mathrm{ins},i}$ and $\Phi_{\mathrm{del},i}$ are defined only for nucleotide positions i of the root sequence, reflecting that genome editing occurs only for predefined targeting sequences in cell lineage tracing.

When the distributed computing mode is enabled to simulate mutations for the template PU tree of n leaves, PRESUME first simulates mutations for its most upstream tree that subdivides its downstream trees so that each of them is composed of $\lfloor \sqrt{n} \rfloor$ or less leaves. The following downstream mutational processes are independently simulated using distributed computing.

The PRESUME parameters used in all experiments in the present study are listed in Supplementary Table 2.

Evaluation of reconstructed trees.

The accuracy of the reconstructed tree compared to the ground truth or reference tree was measured by 1 – NRFD[23] using the phangorn R package version 2.4.0 (ref. [44]). NRFD represents a fraction of edges each of whose disconnections separates leaf sequences of a tree into two groups that cannot be achieved in the other tree by any edge disconnection. Sequences that are not shared between the two target trees were ignored to calculate the score. The coverage of the reconstructed tree was defined by the proportion of initial input sequences successfully included in the reconstructed tree without dropping out as unresolved

problematic sequences during the FRACTAL iterations. The tree recovery ratio was obtained by multiplying accuracy and coverage.

Simulation of pseudo-evolutionary trees.

The SILVA 16S rRNA reference tree (13,897 sequences after removing redundant sequences) and the GTDB reference trees for bacteria and archaea were used to calibrate the topology balancedness parameter σ for the simulation of pseudo-evolutionary lineages. The GTDB trees were combined through root nodes, and a random subgraph consisting of 13,897 leaves was used for the analysis. Various trees with 13,897 to 16,384 leaves were simulated using PRESUME by changing σ from 0 to 4.0. The topological fits of each simulated lineage to the two reference trees were, respectively, measured by similarity in BBI distributions (intersection over union with a binning size of 0.05), where a BBI was calculated for each node by taking the ratio of the number of leaves associated with its two downstream branches (smaller over larger). In this analysis, BBIs were calculated for nodes with more than ten leaves associated with both of their two branches. Separately, Q and a of the GTR-Gamma model and branch lengths of the SILVA 23S rRNA tree (1,614 sequences) were estimated using RAxML (Q* and a*). Although the size of the 23S rRNA tree was onefold smaller than that of the 16S rRNA tree, the 23S rRNA tree composed of longer sequences (median of 2,900 bp) was assumed to produce more relevant mutational process parameters than the 16S rRNA tree (median of 1,447 bp). Applying the optimal topology balancedness parameter σ* that showed the best fit to the 16S rRNA tree (more skewed than GTDB) to PRESUME, a template tree for 1,019,509 sequences was generated, and the relative lengths of the internal and terminal branches were redefined by randomly assigning those of the 23S rRNA tree estimated using RAxML, respectively. Using the GTR-Gamma model with Q* and a*, nucleotide substitution processes were simulated along with the branch-adjusted tree by changing the mutational parameter μ from 0.001 to 1. Finally, the optimal mutational parameter for the pseudo-evolutionary lineage μ* was chosen so that the distribution of NHDs between 1,000 pairs of sequences randomly chosen from the simulated dataset have the best agreement with that of the SILVA 23S rRNA dataset (intersection over union with a binning size of 0.05). Four pseudo-evolutionary lineages were produced by repeating the simulation of nucleotide substitution processes using the same length-adjusted template tree of 1,019,509 sequences and with the above-mentioned optimized parameters (σ*, Q*, a* and μ*).

Identification of scGESTALT barcode arrays.

To determine a representative scGESTALT barcode array for each single cell, the raw FASTQ files were first converted into FASTA format using SeqKit (version 0.10.1). Sequence reads associated with single cells that had only a single unique molecular identifier (UMI) were removed. For each of the remaining single cells, barcode array reads of the same UMIs were then grouped, and the reads in each UMI group were clustered by CD-HIT-EST 4.8.1 (ref. [45]) with the options '-c 0.9' and '-n 7.' Reads in the most dominant cluster with a size ≥75% in each UMI group were aligned using MSA using MAFFT version 7.453, and their consensus sequence was determined by CONS in EMBOSS:6.6.0.0 (ref. [46]). The UMI groups with no dominant cluster of ≥75% were eliminated for the following process. The consensus barcode array reads of different UMIs in each cell were then

processed by the same operations (clustering, MSA and consensus sequence construction) to determine the representative barcode arrays of cells.

Mutation call.

A common mutation-calling pipeline was used throughout the study for the barcode arrays of the scGESTALT dataset and simulated cell populations as well as the EP-PCR reads. Sequences were aligned to the reference (root) sequence by NEEDLEALL in EMBOSS:6.6.0.0, with the options '-gapextend 0.5', '-gapopen 10' and '-endopen 10.' The resulting SAM file was processed using SAMTOOLS version 1.9–74-gf69e678 (ref. [47]) to obtain MD tags representing mutation patterns.

Reconstruction of scGESTALT lineages.

The scGESTALT cell lineages were reconstructed as previously described[9]. Among the identified mutations in the scGESTALT barcode arrays of single cells, those overlapping on the Cas9 cut sites, –3 bp to –1 bp from the 5′ end of the protospacer adjacent motif (PAM), were used for lineage reconstruction. For each lineage reconstruction, a presence/absence matrix for all of the observed unique mutations was generated for every cell, and every mutation was weighted by its relative log abundance across the cells. Using these datasets, a maximum parsimonious cell lineage was then reconstructed using PHYLIP Mix 3.696 (ref. [48]) with the options P, W, 2, 3, 4 and 5 and a maxtree parameter of 5. Among multiple trees reconstructed using Mix, the tree with the highest parsimony score was selected. If there were multiple highest-scored trees, the most common one, if any, was selected. Otherwise, selection was random.

Simulation of high-capacity CRISPR barcode system.

In the simulation of cell proliferation with the high-capacity CRISPR barcode system, each cell was modeled to encode 100 copies of a transcribing barcode array in its chromosomes with specific locus IDs, in which a dual-function base editor Target-ACEmax continuously induces mutations only in the barcode arrays. The design of the barcode array was adopted from scGESTALT, where each root barcode array was composed of a total length of 255 bp, starting from a leader sequence (12 bp) followed by nine targeting barcode units, each of which was composed of a 3-bp linker and a single gRNA target site (24 bp in total; a spacer sequence of 21 bp plus 5′-NGG-3′ PAM for SpCas9) (Supplementary Data 2). Each of the gRNA-targeting spacers was designed to encode adenine or cytosine in every position from –21 bp to –7 bp from the 5′ end of the PAM with equal probabilities to serve them as substrates for C→T and A→G base editing. The cell proliferation processes were simulated using a topology balancedness parameter σ of 0.05.

Base editing.—The stochastic base substitution by Target-ACEmax on every barcode array copy per generation was modeled as subsequently detailed. From the base editing outcome data of Target-ACEmax at 47 gRNA target sites in the human genome, we first calculated the average C→T and A→G editing frequencies per gRNA-targeting site for positions from –25 bp to +5 bp from the PAM and obtained a unit substitution event probability matrix $\dot{\Phi}_{\mathrm{sub}}$ (0 was given for the other substitution probabilities) (Supplementary

Data 3). We then fit a base editing scaling factor χ for $\chi\dot{\boldsymbol{\Phi}}_{\text{sub}}$ to best represent the base editing spectra of each genomic target site and obtained the base editing scaling factor distribution $P(\chi)$ (Supplementary Data 4). Finally, the substitution probability matrix of the entire barcode array per generation $\boldsymbol{\Phi}_{\text{sub}}$ was given such that the substitution probability matrix $\boldsymbol{\Phi}_{\text{sub}}(\tau)$ for each of the nine gRNA target sites $\boldsymbol{\tau}$ (from −24 bp to −1 bp) satisfies the following:

$$\boldsymbol{\Phi}_{\text{sub}}(\tau) = \theta_{\text{sub}}\chi(\tau)\dot{\boldsymbol{\Phi}}_{\text{sub}}$$

where $\chi_\tau \sim P(\chi)$, and $\boldsymbol{\Phi}_{\text{sub}}$ is the secondary base editing scaling factor common across the target sites.

Indels.—The indel introduction process in the tandem array of gRNA target sites by Cas9 base editors was modeled as follows. From the indels observed in the scGESTALT dataset, we first modeled frequencies of observed insert positions and deletion center positions in the root sequence coordinates and obtained the unit insertion event probabilities across different positions, $\dot{\boldsymbol{\Phi}}_{\text{ins}}$, and obtained unit deletion event probabilities across different positions, $\dot{\boldsymbol{\Phi}}_{\text{del}}$, in addition to their size probability distributions $\psi_{\text{ins}}(L)$ and $\psi_{\text{del}}(L)$, respectively, where L is the insertion or deletion length. The insertion event probabilities were given to the $5'$-side nucleotide positions of the observed inserts, and the deletion event probabilities were given to positions each given by $\lfloor (x+y)/2 \rfloor$, where deletion start and end positions in the reference sequence were x and y, respectively. The insertion and deletion event probabilities across positions of the barcode array per generation $\boldsymbol{\Phi}_{\text{ins}}$ and $\boldsymbol{\Phi}_{\text{del}}$ for PRESUME were then defined as follows:

$$\boldsymbol{\Phi}_{\text{ins}} = \theta_{\text{ins}}\dot{\boldsymbol{\Phi}}_{\text{ins}}$$

$$\boldsymbol{\Phi}_{\text{del}} = \theta_{\text{del}}\dot{\boldsymbol{\Phi}}_{\text{del}}$$

where θ_{ins} and θ_{del} are constant scaling factors. The size probability distributions $\psi_{\text{ins}}(L)$ and $\psi_{\text{del}}(L)$ were directly given to PRESUME to simulate indel size at each event.

Simulation of 16 million cells.—A tree topology of 16,671,840 cells was first obtained using PRESUME. Indel accumulation processes were tested for a small tree of 4,000 cells by sliding $\boldsymbol{\Phi}_{\text{ins}}$ and $\boldsymbol{\Phi}_{\text{del}}$ separately each from 0.0001 to 1. The optimal parameters θ^*_{ins} (0.015) and θ^*_{del} (0.34) were calculated. These parameters, together with the simulated cell proliferation trajectories, conferred indels in each barcode arrays whose lengths were similar to those observed in the scGESTALT dataset. To have similar indel levels in each barcode array through the trajectories of generating 16 million cells, we used $\theta^*_{\text{ins}}/2$ and $\theta^*_{\text{del}}/2$, assuming the average total generation time from the root to leaves of a given tree was double to that of a tree with a square root number of leaves. To determine secondary base editing

scaling factors Φ_{sub} to be used for the simulation of 16 million cells, sequence diversification processes were also simulated along with the same tree of 4,000 cells with the calibrated indel scaling parameters θ^*_{ins} and θ^*_{del} by sliding Φ_{sub} from 0.001 to 10. Elements of the matrix Φ_{sub} for each nucleotide position (either C→T or A→G) were fixed at 1 when they exceeded 1. In each simulation, the produced barcode arrays with mutations were aligned to the reference (root) sequence for calling mutations as described above. Substitutions in different numbers of barcode arrays (5, 10, 20, 40 and 100) were used for cell lineage reconstruction by RapidNJ and FRACTALized RapidNJ. From this experiment, the optimal secondary base editing factor θ^*_{sub} (0.5) that showed that the overall best accuracy was determined to reconstruct the lineage of 4,000 cells with any number of barcode arrays. Finally, together with the defined indel scaling parameters, the predicted optimal secondary base editing scaling factor $\theta^*_{sub}/2$ (0.25) was used to simulate a proliferation of 16 million cells. Another simulation with $\theta^*_{sub}/20$ (0.025) was also performed.

Cell lineage tracing.

To reconstruct the lineage of 16 million cells using FRACTALized RapidNJ, we aligned the mutated barcode array sequences of each cell to the reference sequence, called mutations and used only substitutions. To examine the performance of different lineage estimation methods, cells from different sizes of subclades in the simulated lineages of 16,671,840 cells were used for cell lineage tracing using the original Cassiopeia (version 0.0.1), FRACTALized Cassiopeia, FRACTALized RapidNJ and FRACTALized RAxML. For Cassiopeia and FRACTALized Cassiopeia, the barcode array sequences were first aligned to the reference (root) sequence, and the identified mutations (both indels and substitutions) were grouped separately for the leader sequence and nine gRNA target units, where deletions spanning across multiple groups were assigned to all of the corresponding groups, and the converted mutation lists were formatted by ‘utilities.alleletable_to_character_matrix’ of the Cassiopeia package. The lineage reconstruction experiments by Cassiopeia were performed using the three run modes ‘Greedy’, ‘Hybrid’ and ‘ILP’ with the options ‘–greedy–num_threads 1’,’– hybrid–num_threads 1–max_neighborhood_size 6000–time_limit 5000–cutoff 200’ and ‘–ilp–num_threads 1–time_limit 12600’, respectively, as described previously. Lineage reconstruction experiments using FRACTALized Cassiopeia were also performed with the three run modes with the options ‘–greedy–num_ threads 1’, ‘–hybrid–num_threads 1–time_limit 60–cutoff 4’ and ‘–ilp–num_ threads 1–time_limit 60’, respectively, with the subsampling size and the naive computing threshold set to 10. For FRACTALized RapidNJ and RAxML, to realize a fair comparison with Cassiopeia and FRACTALized Cassiopeia, the grouped mutation lists of each cell are concatenated into one list to include the redundant deletions assigned to multiple groups if any. The mutation lists of cells were then given to FRACTALized RapidNJ and RAxML with the above-mentioned mode, taking into account both indels and substitutions. The FRACTAL parameters used in these experiments are listed in Supplementary Table 1.

PCR-based sequence evolution.

First-round ER-PCR and fragment cloning.—The *CTNNB1* gene coding region (198 bp; chromosome 3, 41246920–41247074) and BET002 non-coding region (110 bp; chromosome 22, 27202644–27202753) in the human genome GRCh37/hg19 were amplified by EP-PCR using the GeneMorph II Random Mutagenesis kit (Agilent) with primer pair 1684–1827 for *CTNNB1* and 1759–1760 for BET002, respectively (Supplementary Table 3). EP-PCR was performed in a 10-μl reaction volume, including approximately 10 copies of human genomic DNA template, 0.5 μl of 10 μM primer each, 0.2 μl of Mutazyme II DNA polymerase, 1 μl of 10× Mutazyme II Reaction Buffer and 0.2 μl of 40 mM dNTPs with the following thermal cycler conditions: 95 °C for 120 s, 60 cycles of 95 °C for 30 s, 55 °C for 30 s and 72 °C for 60 s and 72 °C for 10 min for the final extension. For low copy number genomic DNA template preparation, 3.45 ng (approximately 1,000 copies) of human genomic DNA was resolved in 100 μl of PCR water containing 100 ng of pRSI9-U6-(sh)-UbiC-TagRFP-2A-Puro plasmid DNA (Addgene, 28289) as a decoy in a 1.5-ml LoBind tube (Eppendorf) using 1 μl each as the PCR template. Amplified PCR products were resolved by 2% agarose gel electrophoresis, purified using a column purification kit (NipponGene) and digested with XbaI (New England Biolabs) and XhoI (New England Biolabs). The digested products were purified again and assembled with the XbaI-XhoI-digested pcDNA3.1(+) backbone plasmid using T4 DNA ligase (NipponGene) at 16 °C for 2 h. The assembly product was purified using a 1.8×volume of Agencourt AMPureXP (Beckman Coulter) and transformed into New England Biolabs 5α *Escherichia coli* chemically competent cells according to the manufacturer's high-efficiency transformation protocol (New England Biolabs). Bacterial colonies were isolated, and the target sequence region was amplified by direct colony PCR using primer pairs 471–1727 for *CTNNB1* and 471–1760 for BET002 (Supplementary Table 3). For PCR samples in an expected size range, the corresponding plasmids were purified, and the mutations were confirmed by Sanger sequencing using primer 471 (Supplementary Table 3). For *CTNNB1* and BET002, we prepared a collection of 96 colony-isolated plasmid samples, which were observed to have mutations via Sanger sequencing.

Second-round PCRs.—For *CTNNB1* and BET002, the first-round plasmid products were assembled in a 96-well round-bottom deep-well plate at a concentration of approximately 100 copies per μl. Each sample was then subjected to second-round EP-PCR and HF-PCR. Each second-round EP-PCR was performed using the same protocol as the first-round EP-PCR using 1 μl of template and primer pair 1788–1789 for *CTNNB1* or 1783–1784 for BET002 to attach common adapter sequences for the following indexing PCR (Supplementary Table 3). The PCR products were resolved by 2% agarose gel electrophoresis and purified using a 1.8× volume of Agencourt AMPureXP (Beckman Coulter). Each HF-PCR also with the primer pair 1788–1789 for *CTNNB1* or 1783–1784 for BET002 was performed in a 20-μl reaction volume, including 1 ng of template plasmid, 0.5 μl of 20 μM each primer, 0.2 μl of Phusion DNA Polymerase, 4 μl of 5× Phusion HF Buffer, 2 μl of 2 mM dNTPs and 0.06 μl of 100-fold SYBR Green I dilution (for monitoring amplification signal in each well) with the following thermal cycle conditions: 98 °C for 30 s, 30 cycles of 98 °C for 10 s, 60 °C for 10 s and 72 °C for 60 s and 72 °C for 5 min for the final extension.

Preparation of sequencing library.—The adapter-tagged PCR amplicon products were diluted 20-fold using PCR water for library preparation. To generate a pooled sequencing library, index PCR was carried out to reamplify each of the second-round EP-PCR and HF-PCR products using a pair of custom Illumina 9-bp index primers as described previously[32] (Supplementary Table 4). In brief, 1 μl of the template DNA was amplified in a 10-μl reaction volume, including 1 μl of 10 μM primer each, 0.2 μl of Phusion DNA Polymerase, 2 μl of 5×Phusion HF Buffer and 1 μl of 2.5 mM dNTPs with the following thermal cycle conditions: 98 °C for 30 s, 15 cycles of 98 °C for 10 s, 60 °C for 10 s and 72 °C for 60 s and 72 °C for 5 min for the final extension. For *CTNNB1* and BET002, the final products were resolved by 2% agarose gel electrophoresis and purified using a gel-based purification kit (NipponGene). The EP-PCR and HF-PCR products were combined into single tubes. Each pooled sequencing library was quantified using the Library Quantification kit (KAPA Biosystems) and sequenced by MiSeq v.3 2 × 150 bp paired-end sequencing (Illumina) with 20% PhiX spike-in control (Illumina). FASTQ files were generated using bcl2fastq2 (v.2.20.0). The second-round EP-PCR experiment was performed in duplicate, each of which was sequenced independently. FASTQ files for duplicates were combined for subsequent analysis.

Read preprocessing.—After demultiplexing of the sample reads, the adapter and sample index sequences were removed from the reads. The sequencing reads were then filtered with a Q score threshold of ≥5 for every position and an average Q score threshold of ≥20 per read.

Lineage estimation of EP-PCR-diversified sequences.

First-generation EP-PCR-diversified sequences were reconstructed from the HF-PCR reads of the corresponding reaction wells. Sequencing reads of each reaction well were processed by BARTENDER_SINGEL_COM of bartender-1.1 (ref. [49]) with '-c 10 -l 5 -s 1 -z 3 -d 5' to correct sequencing and PCR errors. Unique sequencing reads were then counted to generate a rank-count plot. The knee point of the rank-count plot was determined by KNEED version 0.5.0 (ref. [50]) and set as a threshold to determine the first-generation sequences and their abundances in each reaction well. Unique reads of the second-round EP-PCR were used as the second-generation EP-PCR-diversified sequences of each reaction well. For *CTNNB1* and BET002, the second-generation sequences of all reaction wells were pooled, and their entire lineage was reconstructed using FRACTALized RAxML (Supplementary Table 1). After lineage reconstruction, assuming second-generation sequences that best matched the first-generation sequences of different wells to be from PCR artifacts or contamination, the Levenshtein distances[51] were measured between the first- and second-generation sequences. Second-generation sequences harboring unique parental first-generation sequences in their corresponding reaction wells were identified. Sequences that did not meet this criterion were excluded from the following analyses.

Normalized distance of tree leaves.

To evaluate the distance of two second-generation EP-PCR-diversified sequences in the reconstructed tree, we simply introduced a metric of 'normalized clade size', where distance $\widetilde{D}_{\mathrm{AB}}$ of two leaves A and B in a tree T was calculated by the number of leaves in a subclade

M_{AB} starting from their most recent common ancestor normalized by the number of entire leaves in the tree:

$$\widetilde{D}_{AB} = \frac{\text{Number of leaves in } M_{AB}}{\text{Number of leaves in T}}.$$

To evaluate the difference between intra- and interwell distances of sequences in the tree, we randomly sampled 1,000 second-generation sequences for each reaction well if there were 1,000 or more sequences. Otherwise, the entire sequences were sampled. We then extracted a subtree composed of the sampled sequences from the entire tree. Using the extracted subtree, the intrawell distances of each target well were measured for 1,000 pairs of randomly selected sequences. Control interwell distances for the target well were measured for 1,000 pairs of sequences, with one randomly selected from the target well and the other selected randomly from a random well. For each target well, the difference between the intrawell distance distribution and the control interwell distance distribution was tested by the Brunner–Munzel test using the R package Brunner–Munzel 1.4.1 (ref. [52]) followed by Bonferroni correction.

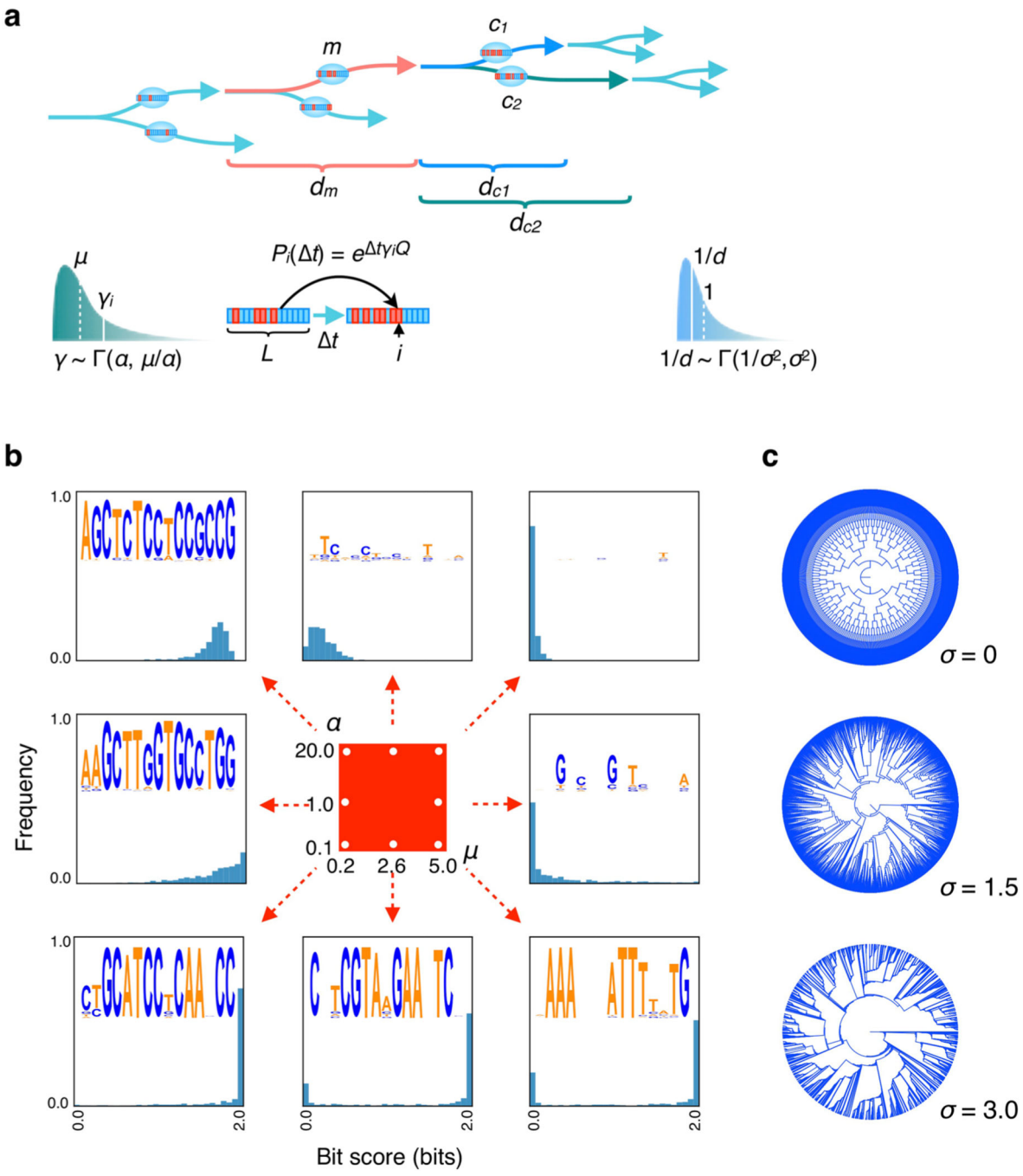
Distributed computing environment.

All of the lineage estimations using FRACTAL and sequence diversification simulations using PRESUME were performed with the SHIROKANE Supercomputer at the University of Tokyo Human Genome Center (Xeon E5–2670 v.3) or the NIG Supercomputer System at the National Institute of Genetics (AMD EPYC7501, Xeon Gold 6130 or Xeon Gold 6136). See Supplementary Tables 1 and 2 for further details.

Reporting Summary.

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

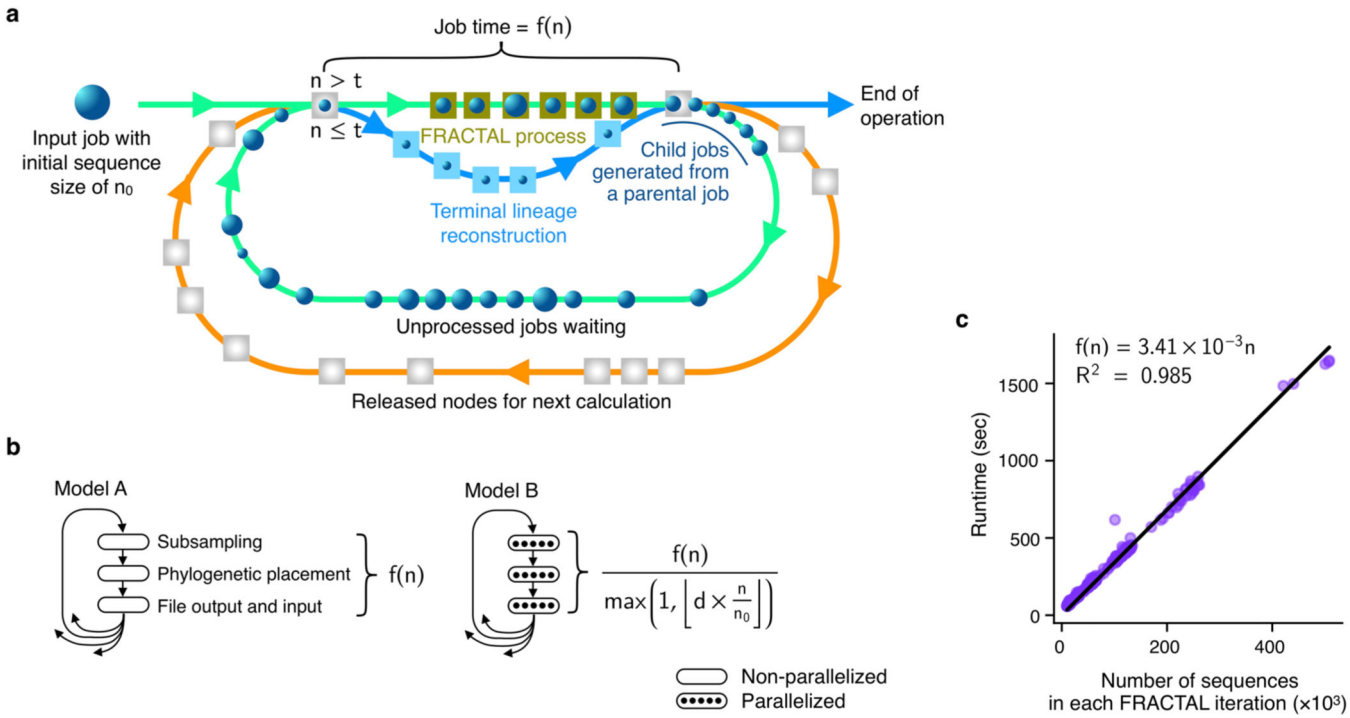
Extended Data



Extended Data Fig. 1 |. PRESUME.

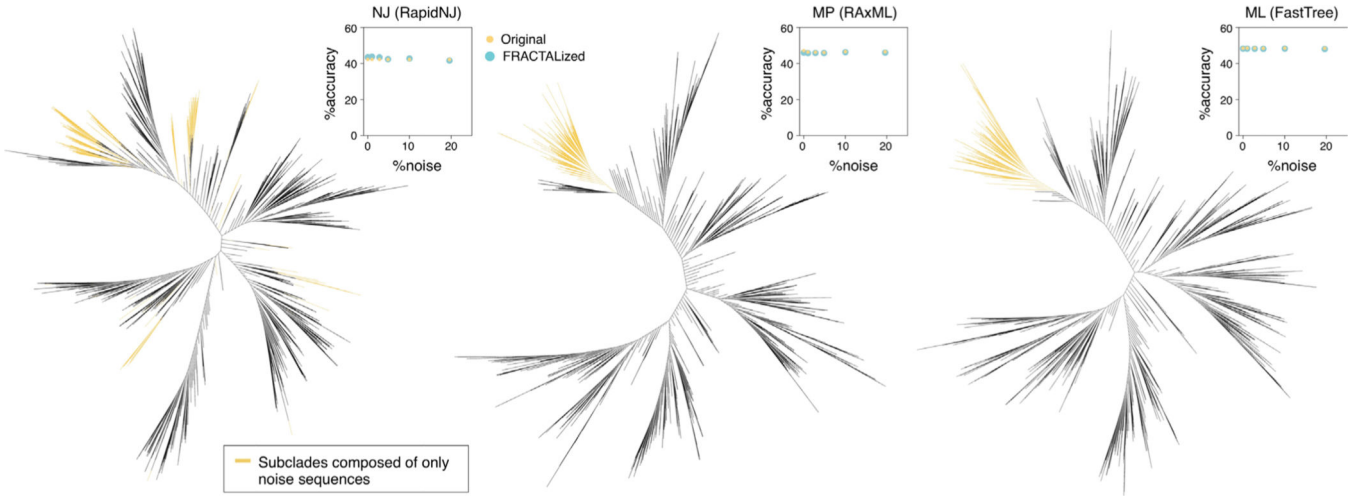
a, Schematic diagram of PRESUME. **b**, Various datasets of 32,768 sequences generated by PRESUME with different mutational parameters μ and α. The topological parameter σ was fixed to 0 (perfectly balanced sequence diversification). The diversity of nucleotide letters across different sequence positions is represented by a bit score distribution and

by a sequence logo for the first 15 nt of the generated sequences. **c**, Partial diagrams of representative trees generated by PRESUME with different topological parameters σ.



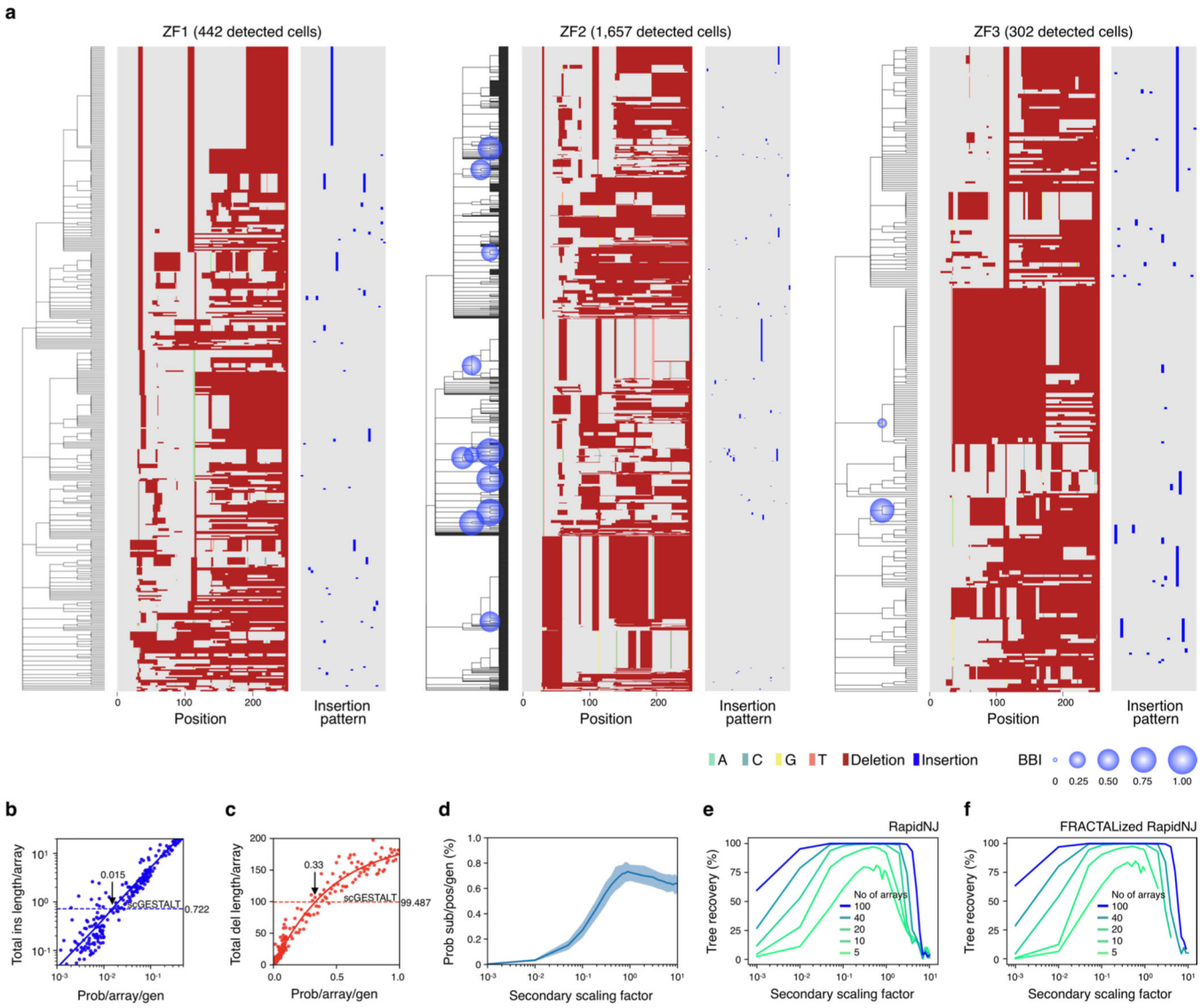
Extended Data Fig. 2 |. Runtime simulation of FRACTAL.

a, Conceptual diagram of the runtime simulation of FRACTAL. Under a distributed computing environment with a fixed number of available nodes d, each FRACTAL job of input sequence size n starts if there is one or more available free computing node; otherwise, it is stalled until a node is released from one of the ongoing jobs. Each FRACTAL job process is modeled to occupy one computing node for a runtime $f(n)$ and produce two new child jobs for the next job cycles each with an input sequence size of $\lfloor n/2 \rfloor$. When the input sequence size is under a certain threshold ($n \leq t$), the job terminates after occupying one node for a runtime $f(n)$ assuming the terminal lineage reconstruction process. **b**, Two implementation models of FRACTAL. In FRACTAL, each job cycle contains three steps that require high computing loads: sequence subsampling from the entire input sequences, phylogenetic placement, and sorting of sequences mapped on each of the sample lineage clades for the next job cycles. In model A, none of the three steps is parallelized where a runtime of each cycle is $f(n)$ for the input sequence size of n. In model B, all of the three steps are perfectly parallelized where a runtime of each cycle is linearly reduced by the number of available computing nodes. The denominator of the formula represents the number of available computing nodes, assuming that the smaller the input sequences for a FRACTAL iteration cycle, the higher the likelihood of computing nodes being occupied by the other job processes at the same period. **c**, Linear regression $f(n)$ using runtime log data of independent FRACTAL iteration cycles any of whose major three steps was not parallelized for the lineage reconstruction of the 235 million sequences.



Extended Data Fig. 3 |. Robustness of evolutionary lineage reconstruction with sequence data noise.

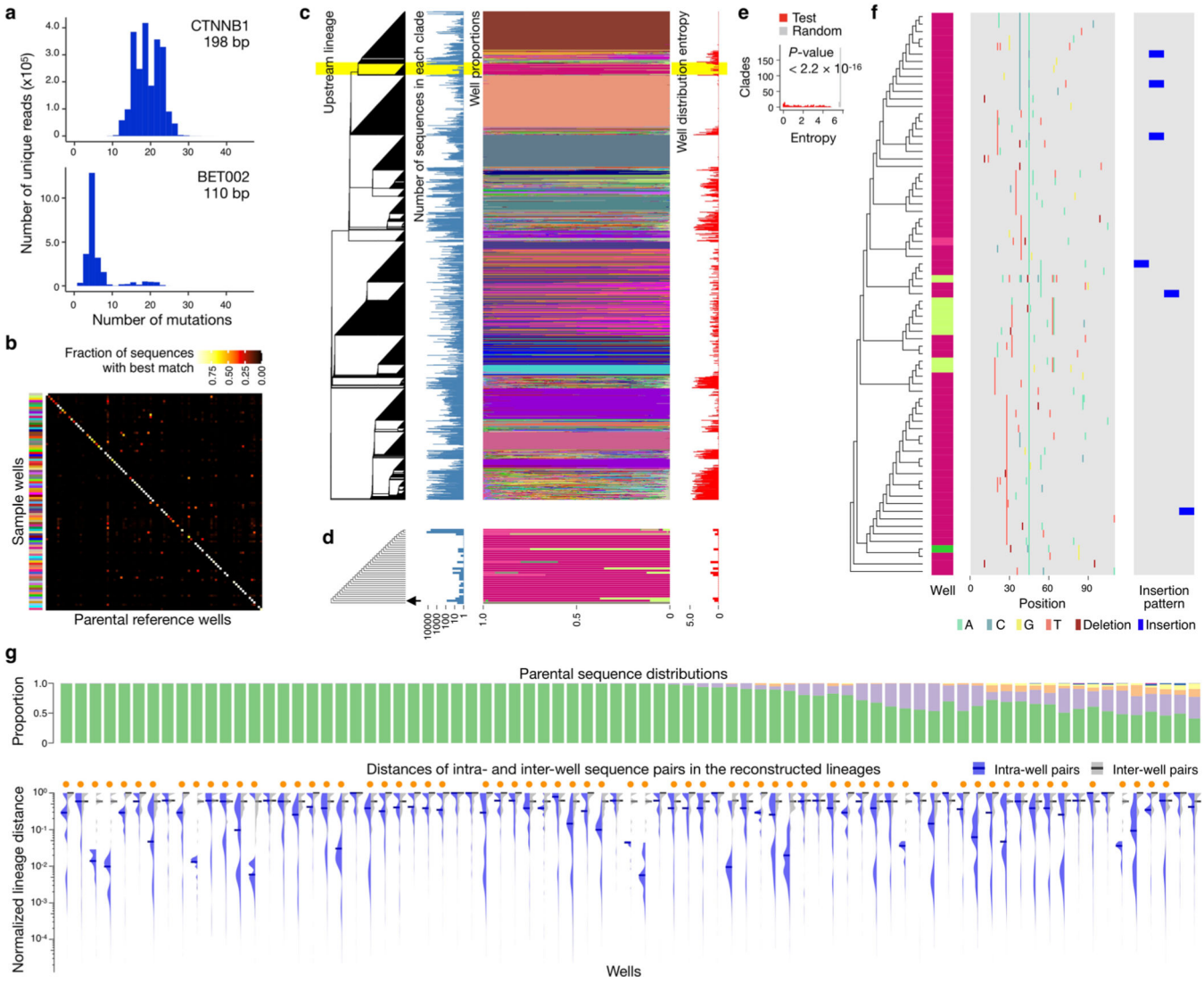
The scatter plots represent accuracies to reconstruct the SILVA 16S rRNA lineage from the input sequence datasets including different fractions of noise sequences. The noise sequences were generated by shuffling nucleotide positions of randomly selected sequences by keeping their alignment gap positions in the multiple sequence alignment result. The lineage dendrograms represent the ones reconstructed from the sequence dataset with 20% noise using FRACTALized RapidNJ, RAxML and FastTree.



Extended Data Fig. 4 |. Simulating proliferation of cells with the scalable CRISPR-barcode system.

a, The scGESTALT lineage tracing datasets of zebrafish brain development. BBI scores were obtained for branches that had more than three leaves associated to each of the two descending edges and a total of more than ten leaves associated to both of the two descending edges. **b, c**, Estimation of the insertion and deletion event probabilities per barcode array per generation in the scGESTALT dataset. The relative insertion and deletion event probabilities across each barcode array position and the probability distributions of insertion and deletion sizes were modeled using those observed in the scGESTALT dataset. The probabilities of the insertion and deletion events per generation for the production of 4,000 cells were fitted to the average total insertion and deletion lengths per barcode array observed in the scGESTALT dataset, respectively. **d**, Median fractions of base substitution per nucleotide position per generation observed for different secondary scaling factors for base editing in the simulation of producing 4,000 cells. The blue shading represents the 5

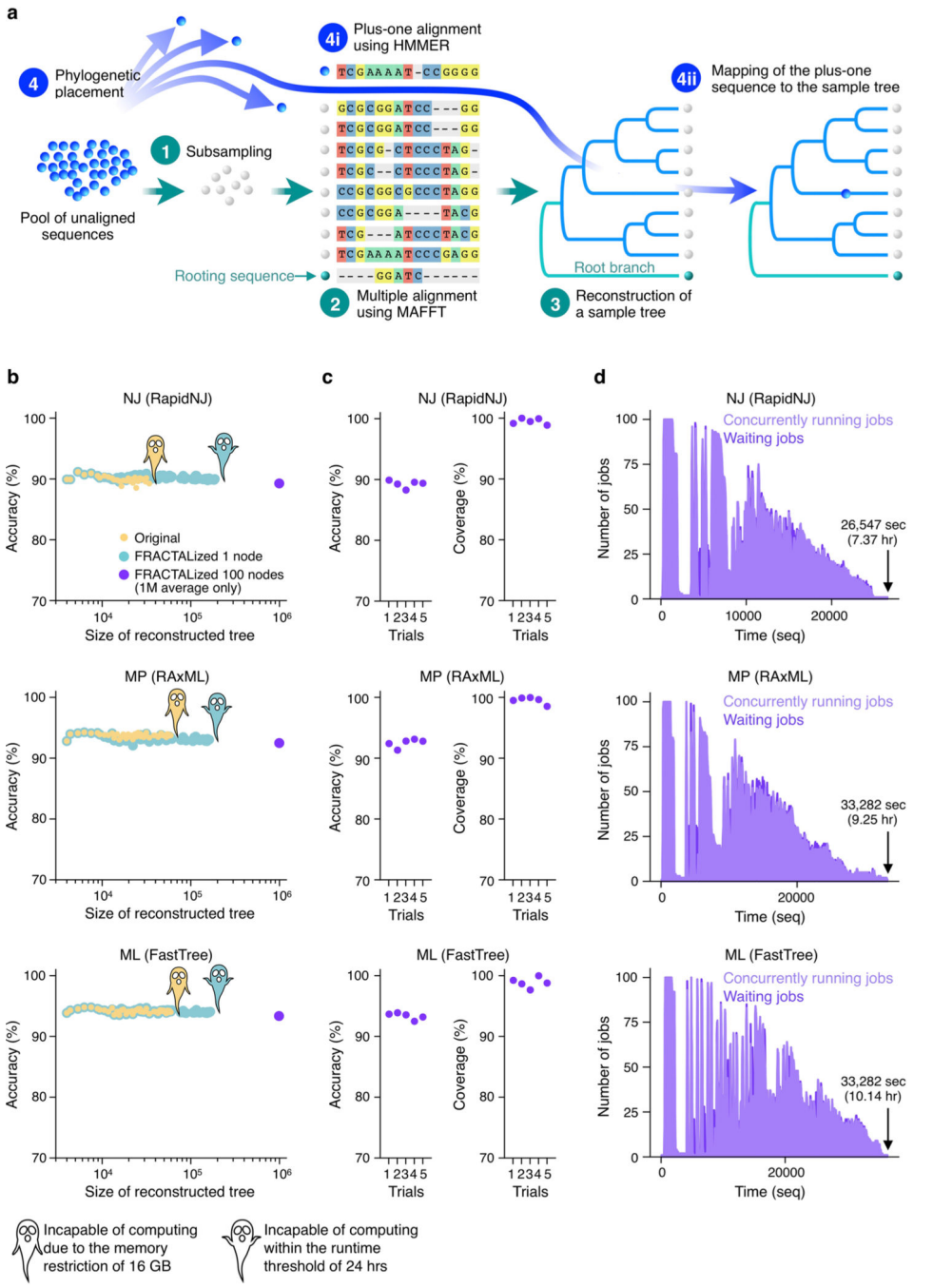
to 95 percentile range. **e, f**, Tree recovery scores for RapidNJ and FRACTALized RapidNJ in reconstructing the lineages of 4,000 cells simulated based on different secondary scaling factors for base editing.



Extended Data Fig. 5 |. Reconstruction of the BeT002 sequence diversification process produced by EP-PCR.

a, Distribution in number of mutations observed in the second-generation sequences (CTNNB1 and BET002). **b-g**, BET002. **b**, Distribution of the second-generation sequences across the parental sample wells. The second-generation sequences were assigned based on their best matched parental first-generation sequences. **c**, The lineage tree of the mutated sequences reconstructed by FRACTAL. The sequences that did not have unique best matched sequences in the expected parental wells were filtered out after the lineage reconstruction. The dendrogram only represents the upstream lineage of the largest clades each composed of less than 15,000 sequences. Number of sequences, proportions of their source sample wells, and entropy of the well proportions are represented for each of the

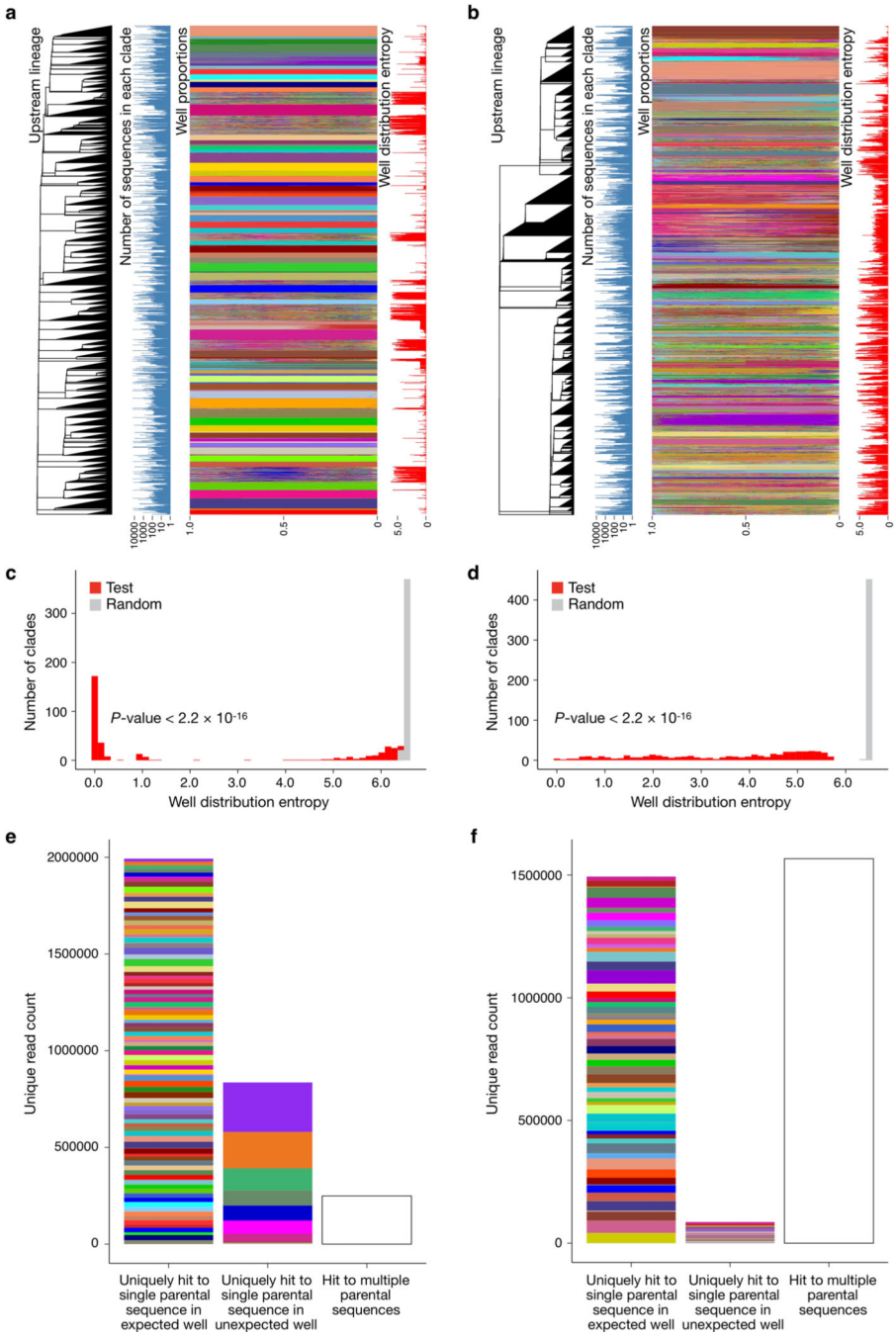
clades. **d**, A zoom-in diagram of the lineage highlighted by yellow in **c**. **e**, Distribution of entropies for the clades each with 1,000 or more sequences. The statistical difference between the entropy distribution and the null distribution given by random sequence-well assignment was tested by two-sided Brunner-Munzel test. **f**, Reconstructed lineage of the sequences in the clade indicated by the arrow in **d**. **g**, Proportions of the parental sequences identified in the control PCR wells and normalized distances of the second-generation sequences in the reconstructed lineage for pairs in the same wells and pairs, one of each is from a different well. Orange dots represent significant differences between the intra- and inter-well distributions (two-sided Brunner-Munzel test with Bonferroni correction; adjusted P-value < 0.05).



Extended Data Fig. 6 |. Evolutionary lineage reconstruction without preliminary MSA.

a, Sample tree reconstruction and phylogenetic placement of FRACTAL with no preliminary multiple sequence alignment (MSA). A given number of sequences are first randomly subsampled from the input sequences (Step 1). The subsampled sequences are aligned with a common root sequence by MSA using MAFFT (Step 2) and a sample tree is reconstructed by a software tool of choice (Step 3). Each of the remaining input sequences are then independently added to the MSA result by 'plus-one' alignment using HMMER (Step 4i) and placed on the sample tree (Step 4ii). **b**, Accuracies of reconstructing various

sizes of clades in the reference lineage of 1,000,000 sequences generated by RNASim. **c**, Accuracies and coverages of reconstructing the entire lineage of 1,000,000 unaligned sequences by FRACTALized RapidNJ, RAxML and FastTree with 100 computing nodes (five trials). **d**, Time series for the numbers of computing jobs and waiting jobs observed in the reconstruction of the simulated lineage of RNA evolution using 100 computing nodes for FRACTALization.



Extended Data Fig. 7 |. Lineages of CTNNB1 and BET002 datasets before filtering out the potential artifact sequences.

a, **b**, Reconstructed lineages. The dendrogram only represents the upstream lineage of the largest clades each composed of less than 15,000 sequences. Number of sequences, proportions of their source sample wells, and entropy of the well proportions are represented for each of the clades. **a**, CTNNB1. **b**, BET002. **c, d**, Distribution of entropies for the clades with 1,000 or more sequences. **c**, CTNNB1. **d**, BET002. The statistical differences between the entropy distributions and the null distributions given by random sequence-well assignment were tested by two-sided Brunner-Munzel test. **e, f**, Unique read counts of the second-generation sequences uniquely best-matched to single parental sequences in the expected and unexpected wells and those best-matched to multiple parental sequences. The ones uniquely best-matched to single parental sequences are color-coded according to the parental wells. The second-generation sequences best-matched to single parental sequences of unexpected wells can be assumed to be cross-contaminants derived during the second EP-PCR and the following steps. The second-generation sequences redundantly best-matched to multiple parental sequences can be assumed to have parental sequences that were either cross-contaminated before the second EP-PCR or were conferred an insufficient number of mutations. **e**, CTNNB1. **f**, BET002.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGEMENTS

We thank members of the Yachie lab for valuable discussions and critical assessment of the work, especially A. Adel, S. King and S. Okawa for reviewing the manuscript. We also thank C. de Hoog for providing comments on the manuscript. Deep sequencing was performed with the support of H. Aburatani. This study was supported by the Canada Research Chair program (by the Canadian Institutes for Health Research), the Japan Science and Technology Agency (JST) PRESTO program (10814), the Japan Agency for Medical Research and Development (AMED) PRIME program (20gm6110007), the Shimadzu Science and Technology Foundation, the Naito Foundation, the Nakajima Foundation and the Asahi Glass Foundation (all to N.Y.). Y.K., S.I., H.M. and N.M. were supported by Japan Society for the Promotion of Science (JSPS) Research Fellowships. High-performance computing experiments were performed using the SHIROKANE Supercomputer at the University of Tokyo Human Genome Center or the NIG Supercomputer System at the National Institute of Genetics.

Data availability

All of the simulated sequences and their lineages generated by PRESUME are available via the URLs given in Supplementary Table 2. The high-throughput sequencing data produced in this work are available at the NCBI Sequence Read Archive (PRJNA675984). *P* values obtained in this study are listed in Supplementary Data 5.

References

1. Zou Q, Wan S, Zeng X. & Ma ZS Reconstructing evolutionary trees in parallel for massive sequences. BMC Syst. Biol 11, 100 (2017). [PubMed: 29297337]
2. Mora C, Tittensor DP, Adl S, Simpson AG & Worm B. How many species are there on Earth and in the ocean? PLoS Biol. 9, e1001127 (2011).
3. Cong L. et al. Multiplex genome engineering using CRISPR/Cas systems. Science 339, 819–823 (2013). [PubMed: 23287718]
4. Mali P. et al. RNA-guided human genome engineering via Cas9. Science 339, 823–826 (2013). [PubMed: 23287722]

5. Kalhor R. et al. Developmental barcoding of whole mouse via homing CRISPR. Science 361, eaat9804 (2018).
6. Chan MM et al. Molecular recording of mammalian embryogenesis. Nature 570, 77–82 (2019). [PubMed: 31086336]
7. Bowling S. et al. An engineered CRISPR–Cas9 mouse line for simultaneous readout of lineage histories and gene expression profiles in single cells. Cell 181, 1410–1422 (2020). [PubMed: 32413320]
8. Salvador-Martinez I, Grillo M, Averof M. & Telford MJ Is it possible to reconstruct an accurate cell lineage using CRISPR recorders? eLife 8, e40292 (2019).
9. McKenna A. et al. Whole-organism lineage tracing by combinatorial and cumulative genome editing. Science 353, aaf7907 (2016).
10. Raj B. et al. Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. Nat. Biotechnol 36, 442–450 (2018). [PubMed: 29608178]
11. Spanjaard B. et al. Simultaneous lineage tracing and cell-type identification using CRISPR–Cas9-induced genetic scars. Nat. Biotechnol 36, 469–473 (2018). [PubMed: 29644996]
12. Alemany A, Florescu M, Baron CS, Peterson-Maduro J. & van Oudenaarden A. Whole-organism clone tracing using single-cell sequencing. Nature 556, 108–112 (2018). [PubMed: 29590089]
13. Quinn JJ et al. Single-cell lineages reveal the rates, routes, and drivers of metastasis in cancer xenografts. Science 371, eabc1944 (2021).
14. Simeonov KP et al. Single-cell lineage and transcriptome reconstruction of metastatic cancer reveals selection of aggressive hybrid EMT states. Cancer Cell 39, 1150–1162.e9 (2021). [PubMed: 34115987]
15. Cao J. et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. Science 357, 661–667 (2017). [PubMed: 28818938]
16. Cao J. et al. The single-cell transcriptional landscape of mammalian organogenesis. Nature 566, 496–502 (2019). [PubMed: 30787437]
17. Cao J. et al. A human cell atlas of fetal gene expression. Science 370, eaba7721 (2020).
18. Sender R, Fuchs S. & Milo R. Revised estimates for the number of human and bacteria cells in the body. PLoS Biol. 14, e1002533 (2016).
19. Barbera P. et al. EPA-ng: massively parallel evolutionary placement of genetic sequences. Syst. Biol 68, 365–369 (2019). [PubMed: 30165689]
20. Stamatakis A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics 30, 1312–1313 (2014). [PubMed: 24451623]
21. Simonsen M, Mailund T. & Pedersen CNS in International Workshop on Algorithms in Bioinformatics 113–122 (Springer, 2008).
22. Price MN, Dehal PS & Arkin AP FastTree 2—approximately maximum-likelihood trees for large alignments. PLoS ONE 5, e9490 (2010).
23. Robinson DF & Foulds LR Comparison of phylogenetic trees. Math. Biosci 53, 131–147 (1981).
24. Yarza P. et al. The All-Species Living Tree project: a 16S rRNA-based phylogenetic tree of all sequenced type strains. Syst. Appl. Microbiol 31, 241–250 (2008). [PubMed: 18692976]
25. Parks DH et al. A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life. Nat. Biotechnol 36, 996–1004 (2018). [PubMed: 30148503]
26. Frieda KL et al. Synthetic recording and in situ readout of lineage information in single cells. Nature 541, 107–111 (2017). [PubMed: 27869821]
27. Jones MG et al. Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. Genome Biol. 21, 92 (2020). [PubMed: 32290857]
28. Nishida K. et al. Targeted nucleotide editing using hybrid prokaryotic and vertebrate adaptive immune systems. Science 353, aaf8729 (2016).
29. Hwang B. et al. Lineage tracing using a Cas9-deaminase barcoding system targeting endogenous L1 elements. Nat. Commun 10, 1234 (2019). [PubMed: 30874552]
30. Grünewald J. et al. A dual-deaminase CRISPR base editor enables concurrent adenine and cytosine editing. Nat. Biotechnol 38, 861–864 (2020). [PubMed: 32483364]

31. Zhang X. et al. Dual base editor catalyzes both cytosine and adenine base conversions in human cells. Nat. Biotechnol 38, 856–860 (2020). [PubMed: 32483363]
32. Sakata RC et al. Base editors for simultaneous introduction of C-to-T and A-to-G mutations. Nat. Biotechnol 38, 865–869 (2020). [PubMed: 32483365]
33. Du Z, Santella A, He F, Tiongson M. & Bao Z. De novo inference of systems-level mechanistic models of development from live-imaging-based phenotype analysis. Cell 156, 359–372 (2014). [PubMed: 24439388]
34. Ciccarelli FD et al. Toward automatic reconstruction of a highly resolved tree of life. Science 311, 1283–1287 (2006). [PubMed: 16513982]
35. Brown CT et al. Unusual biology across a group comprising more than 15% of domain Bacteria. Nature 523, 208–211 (2015). [PubMed: 26083755]
36. Poe S. & Swofford DL Taxon sampling revisited. Nature 398, 299–300 (1999). [PubMed: 10192331]
37. Chow KK et al. Imaging cell lineage with a synthetic digital recording system. Science 372, eabb3099 (2021).
38. Klein AM et al. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. Cell 161, 1187–1201 (2015). [PubMed: 26000487]
39. Macosko EZ et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. Cell 161, 1202–1214 (2015). [PubMed: 26000488]
40. Yu MK et al. DDOT: a Swiss army knife for investigating data-driven biological ontologies. Cell Syst. 8, 267–273 (2019). [PubMed: 30878356]
41. Shannon P. et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 13, 2498–2504 (2003). [PubMed: 14597658]

References

42. Capella-Gutiérrez S, Silla-Martínez JM & Gabaldón T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinformatics 25, 1972–1973 (2009). [PubMed: 19505945]
43. Shen W, Le S, Li Y. & Hu F. SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation. PLoS ONE 11, e0163962 (2016).
44. Schliep KP phangorn: phylogenetic analysis in R. Bioinformatics 27, 592–593 (2010). [PubMed: 21169378]
45. Li W. & Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics 22, 1658–1659 (2006). [PubMed: 16731699]
46. Madeira F. et al. The EMBL-EBI search and sequence analysis tools APIs in 2019. Nucleic Acids Res. 47, W636–W641 (2019). [PubMed: 30976793]
47. Danecek P. et al. Twelve years of SAMtools and BCFtools. Gigascience 10, giab008 (2021).
48. Baum BR PHYLIP: phylogeny inference package. Version 3.2. Quarterly Review of Biology 64, 539–541 (1989).
49. Zhao L, Liu Z, Levy SF & Wu S. Bartender: a fast and accurate clustering algorithm to count barcode reads. Bioinformatics 34, 739–747 (2018). [PubMed: 29069318]
50. Satopaa V, Albrecht J, Irwin D. & Raghavan B. in 2011 31st International Conference on Distributed Computing Systems Workshops 166–171 (IEEE, 2011).
51. Levenshtein VI in Soviet Physics Doklady, Vol. 10 707–710 (Doklady Akademii Nauk SSSR, 1966).
52. Brunner E. & Munzel U. The nonparametric Behrens–Fisher problem: asymptotic theory and a small-sample approximation. Biom. J 42, 17–25 (2000).

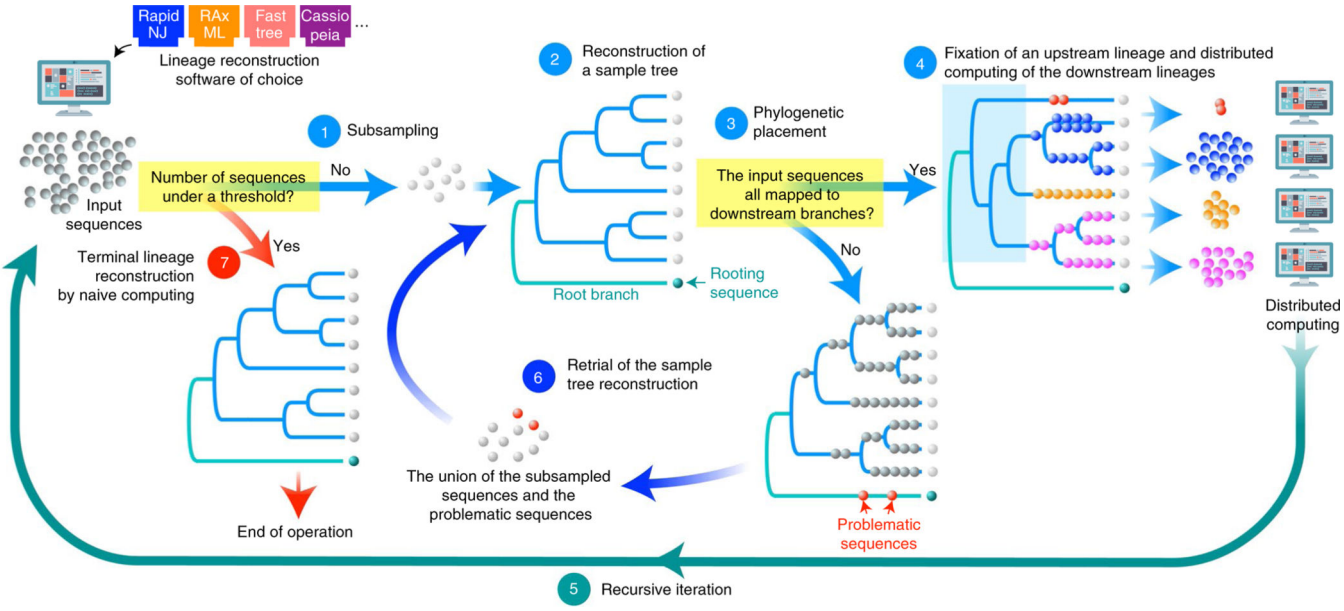


Fig. 1 |.
Schematic representation of FRACTAL.

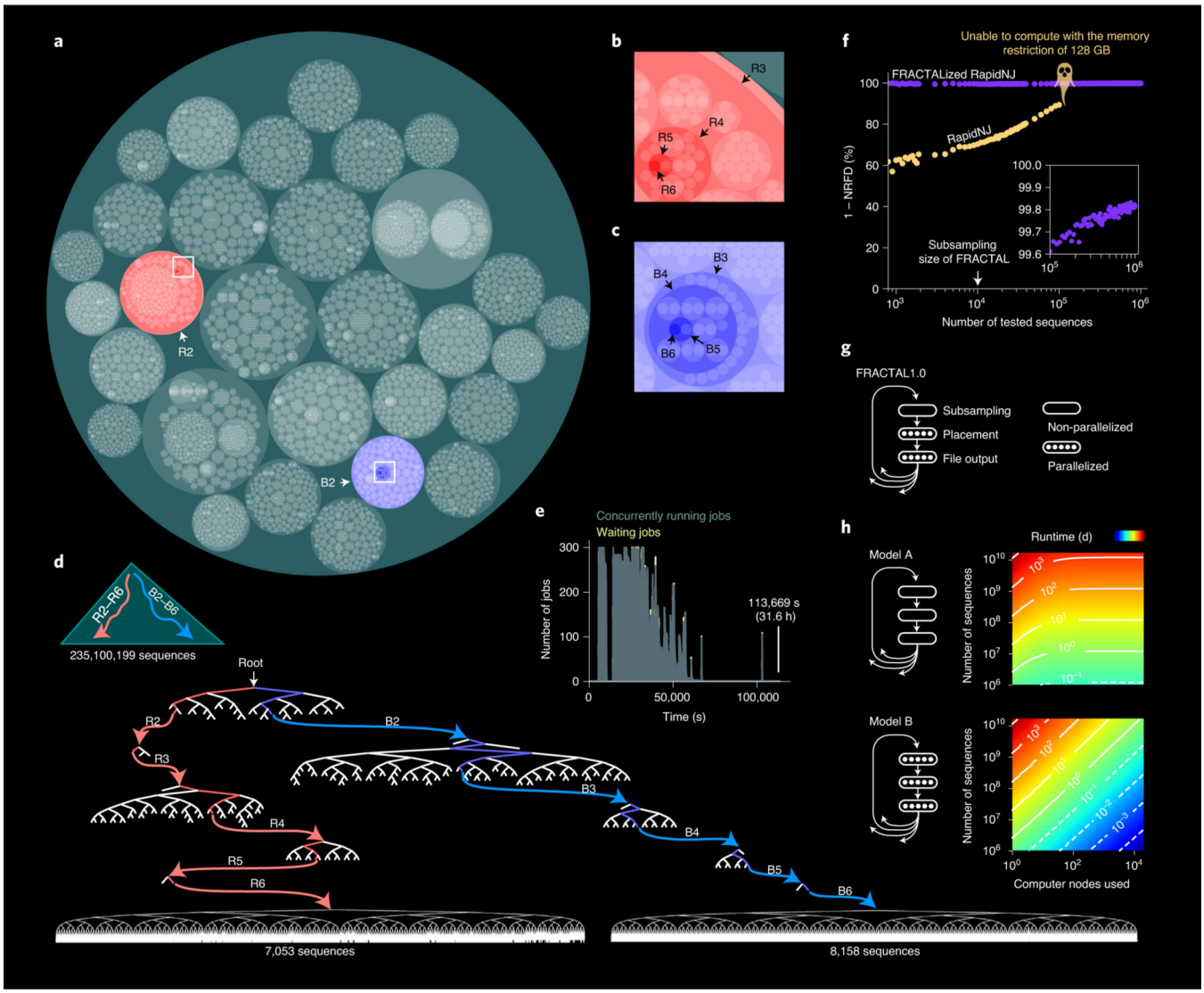


Fig. 2 |. Lineage reconstruction of over 235 million sequences.

a, The whole distributed computing history of FRACTAL used to reconstruct the lineage of 235 million sequences generated by PRESUME. Each circle and its child circles in the circle packing diagram represent a parental FRACTAL iteration cycle and its child job cycles. **b**,**c**, Zoom-in diagrams representing the hierarchies of the distributed computing processes R3–R6 (red circles) (**b**) and B3–B6 (blue circles) (**c**). The hierarchy was visualized using the HiView web application[40]. **d**, A partial representation of the reconstructed lineage of 235,100,199 sequences. Each tree shows a partial lineage determined at each of the distributed computing cycles R2–R6 and B2–B6. The tree diagrams were visualized using Cytoscape 3.7.1 (ref. [41]). Interactive visualization for the whole distributed computing trajectories and lineage subgraphs reconstructed in corresponding FRACTAL cycles are available on the HiView server (http://hiview.ucsd.edu/fractal_235M). **e**, Time series for the numbers of running jobs and waiting jobs observed in FRACTAL computing. **f**, Accuracy estimation of the reconstructed lineage. For different numbers of test sequences randomly sampled from the entire sequences, the agreements between their corresponding subgraphs

in the entire tree reconstructed by FRACTAL and those of the ground truth tree are measured by 1 – NRFD. The accuracies of the tree reconstructed by the original RapidNJ from the same sequence datasets are also shown by 1 – NRFD. The ghost icon denotes the sequence size that failed to reconstruct by the original tool with the memory restriction of 128 GB. **g**, The current implementation of FRACTAL. **h**, Runtime simulations for two potential implementations of FRACTAL. In model A, none of the three major steps in each FRACTAL iteration is parallelized. In model B, all three steps are processed in parallel by another layer of distributed computing.

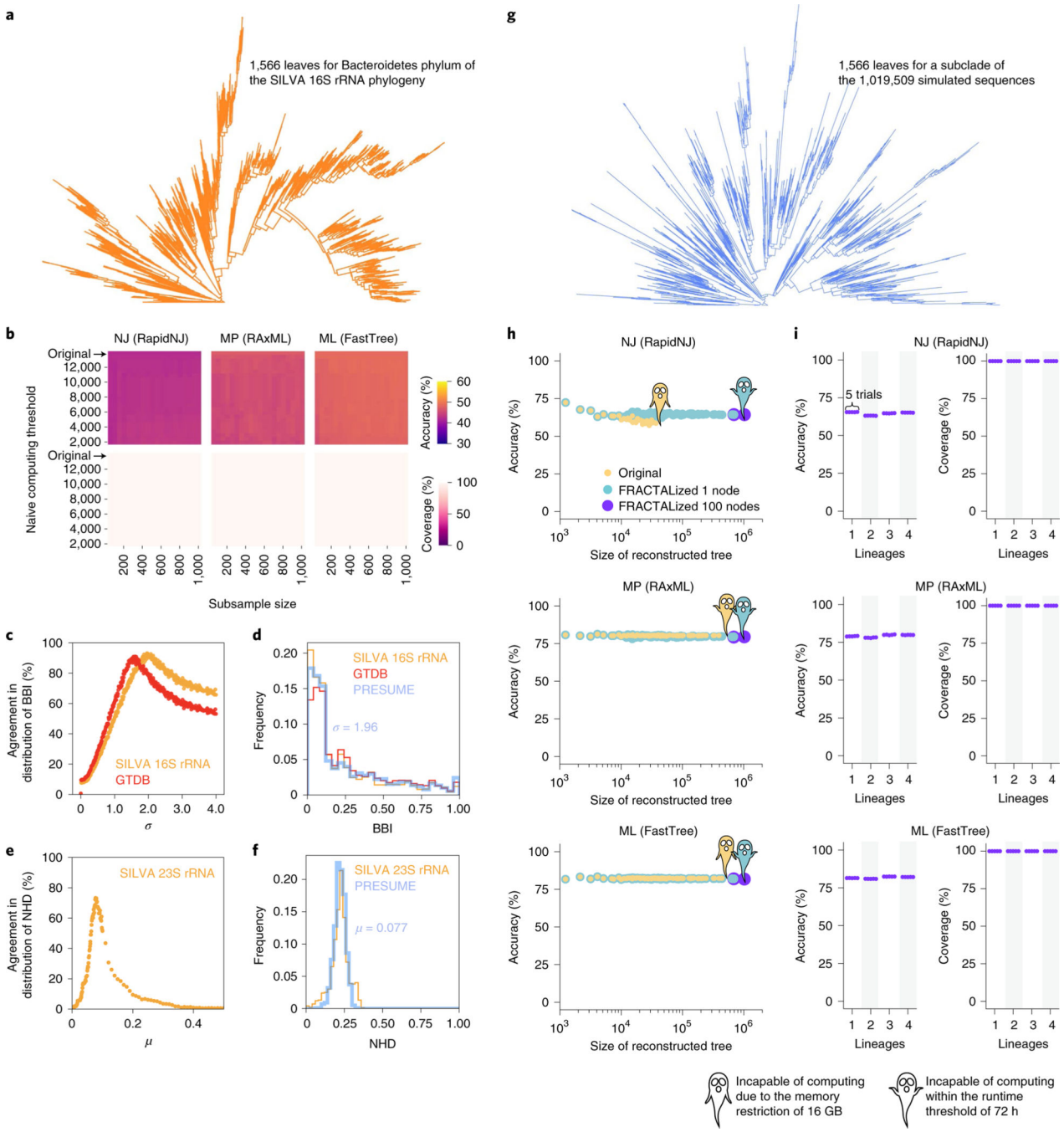


Fig. 3 |. Reconstruction of evolutionary and pseudo-evolutionary trees.

a, A representative subclade of the reference 16S rRNA phylogenetic tree obtained from SILVA. **b**, Accuracies and coverages of reconstructed lineage trees in recapturing the SILVA 16S rRNA reference tree. **c**, Agreements in distribution of BBIs between a reference tree and simulated lineage trees generated with various topological parameters σ. BBI values were calculated for each node by taking the ratio of numbers of leaves associated with its two downstream branches (smaller over larger). The agreement of distributions was measured by intersection over union of two binned histogram areas. **d**, BBI distribution

of the lineage tree fitted the best to the SILVA 16S rRNA reference tree ($\sigma = 1.96$). **e**, Agreements in distribution of normalized Hamming distances (NHDs) among sequences of the SILVA 23S rRNA dataset with those simulated using various mutational parameters μ. **f**, NHD distribution of the simulated dataset fitted the best to the SILVA 23S reference dataset ($\mu = 0.077$). **g**, An example clade of the pseudo-evolutionary lineage of 1,019,509 sequences generated by PRESUME. **h**, Accuracies in reconstructing different sizes of subclades in the pseudo-evolutionary lineage by the original tools and those FRACTALized using single and 100 computing nodes. **i**, Accuracies and coverages of reconstructing four pseudo-evolutionary lineages of 1 million sequences using FRACTAL with 100 computing nodes. Five replicate runs were performed for each of sequence datasets.

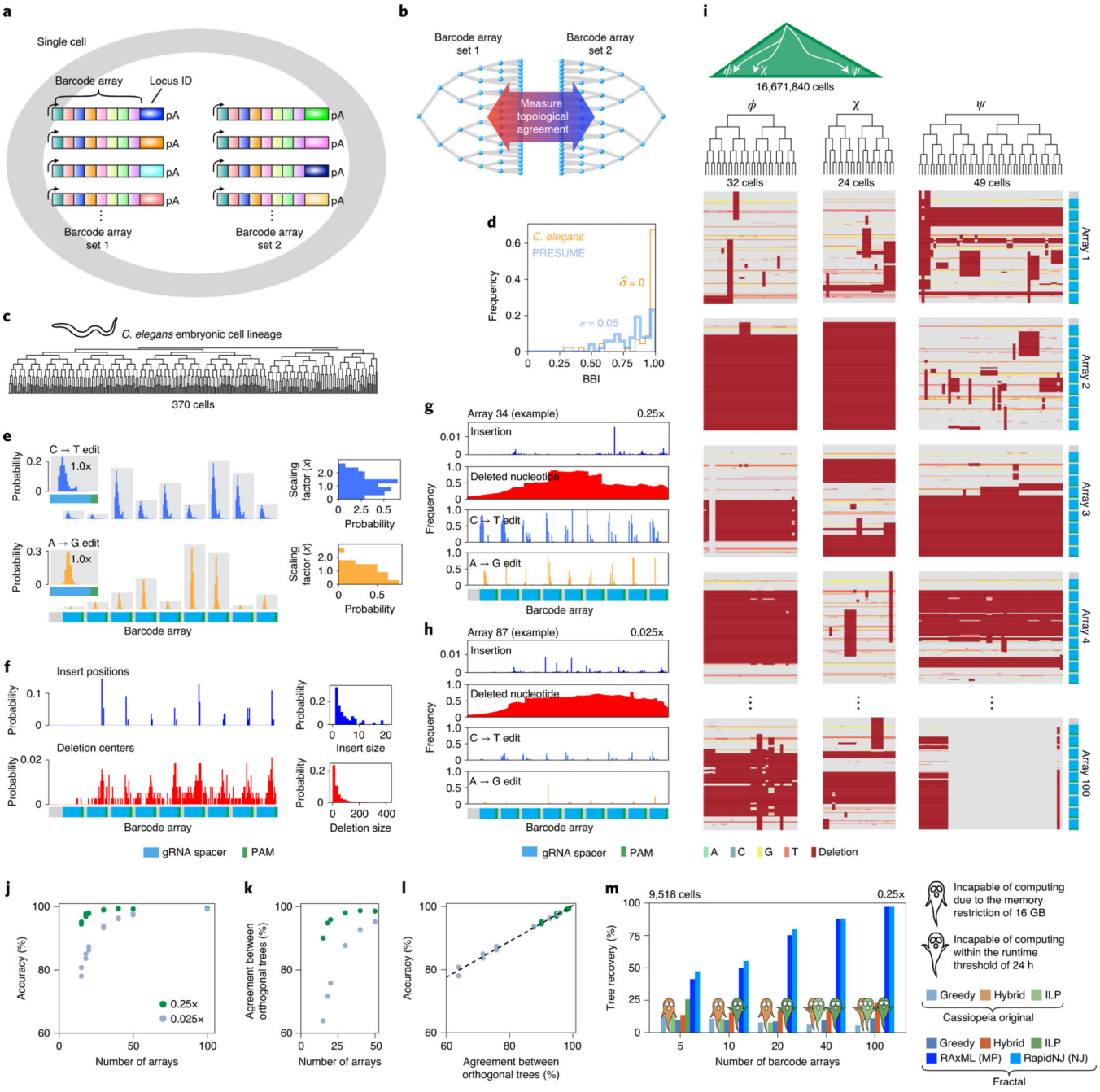


Fig. 4 |. FRACTAL for large-scale cell lineage tracing.

a, A high-resolution CRISPR cell lineage tracing model with a scalable recording capacity. **b**, Validation of the cell lineage tracing method by comparing orthogonally reconstructed lineage trees using independent sets of barcode arrays. **c**, An embryonic cell lineage of *Caenorhabditis elegans*. **d**, BBI distributions of the *C. elegans* cell lineage and a simulated cell lineage tree of the same number of sequences generated with the topological parameter ($\sigma = 0.05$), which was used for the simulation of 16,671,840 cells. **e**, Average C→T and A→G base editing spectra by Target-ACEmax across different positions of a gRNA-targeting unit and distributions of their scaling factors that explain the base

editing frequencies of different target sequences in the previous study. **f**, Probabilities of deletion and insertion events across different positions of the barcode array and their size distributions observed in the scGESTALT dataset. **g**,**h**, Average mutation patterns observed for array 34 in the simulation of 16 million cells using the secondary base editing scaling factor of 0.25 and those observed for array 87 in the simulation of 16 million cells using the secondary base editing scaling factor of 0.025. Base editing frequencies were presented for non-deleted bases. **i**, A partial representation of the lineage of 16 million cells reconstructed using all of the 100 barcode arrays in every cell (the secondary base editing scaling factor of 0.25). Insertions were omitted from the diagram. **j**, Accuracies of the simulated cell lineage reconstructions using different numbers of barcode arrays. **k**, Agreements of orthogonal cell lineage trees for different numbers of barcode arrays used for the orthogonal reconstruction of each tree. **l**, Correlation between lineage reconstruction accuracies using different numbers of barcode arrays and lineage agreements of orthogonally reconstructed trees each by the corresponding numbers of barcode arrays. **m**, Performance comparison of Cassiopeia, FRACTALized Cassiopeia and FRACTALized RAxML and RapidNJ for the lineage reconstruction of 9,518 cells generated using the secondary base editing scaling factor of 0.25. The tree recovery ratio was obtained by multiplying the accuracy and coverage.

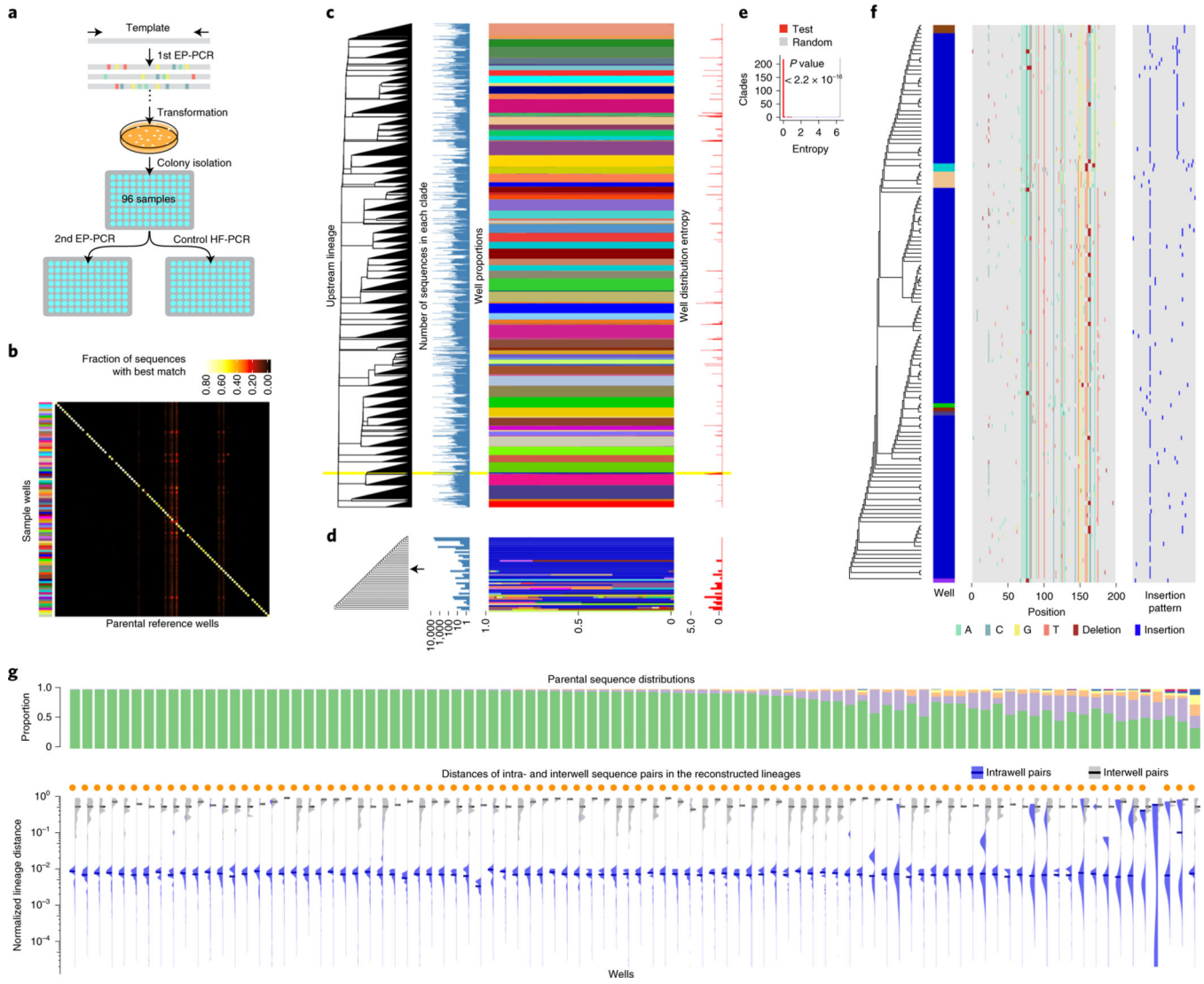


Fig. 5 |. Reconstruction of a sequence diversification process produced by eP-PcR.

a, Experimental design. **b**, Distribution of the second-generation sequences across the parental sample wells. The second-generation sequences were assigned based on their best-matched parental first-generation sequences. **c**, The lineage tree of the mutated sequences reconstructed by FRACTAL. Sequences that did not have unique best-matched sequences in the expected parental wells were filtered out after lineage reconstruction. The dendrogram represents the upstream lineage of the largest clades each composed of less than 15,000 sequences. Number of sequences, proportions of their source sample wells and entropy of the well proportions are represented for each of the clades. **d**, A zoom-in diagram of the lineage highlighted in yellow in **c**. **e**, Distribution of entropies for the clades each with 1,000 or more sequences. The statistical difference between the entropy distribution and the null distribution given by random sequence well assignment was tested by Brunner–Munzel test. **f**, Reconstructed lineage of the sequences in the clade indicated by the arrow in **d**. **g**, Proportions of the parental sequences identified in the control PCR wells and normalized distances of the second-generation sequences in the reconstructed lineage for pairs in the

same wells and pairs, one of each is from a different well. Orange dots represent significant differences between the intra- and interwell distributions (two-sided Brunner–Munzel test with Bonferroni correction; adjusted P value < 0.05).