Theoretical modeling on CRISPR-coded cell lineages: efficient encoding and optimal reconstruction

Ken Sugino[1], Jorge Garcia-Marques[1], Isabel Espinosa-Medina[1], Tzumin Lee[1]

February 6, 2019

[1]Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA, 20147, USA
E-mail: suginok@janelia.hhmi.org, leet@janelia.hhmi.org

Summary

Delineating cell lineages is a prerequisite for understanding the genesis of cell types. Recent studies have demonstrated the feasibility of generating and reconstructing CRISPR/Cas9-coded cell lineages. However, these works have not investigated the limitations or optimality of the encoding or reconstruction processes. Here, we surveyed a multitude of reconstruction algorithms and found hierarchical clustering, with a metric based on the number of shared Cas9 edits, provides the best reconstruction. As to the efficiency, the simple encoding method, with constant Cas9/gRNA edit rate, produces exponential reduction in available coding units and severely limits the trackable depth of lineages. To overcome this, we propose alternative encoding methods, one based on parallel gRNA cascades enabled by CLADES, and another based on variable Cas9 editing rate. Both significantly increase the trackable depth. In summary, we provide a theoretical basis in understanding, designing and analyzing efficient and robust CRISPR-based cell lineage tracking system.

Introduction

Genome-wide single-cell molecular profiling has enabled concurrent detailed analysis of cell identity and cell lineage in complex tissues. The identities of individual cells can be inferred from gene expression patterns, via single-cell RNA sequencing. As to cells' lineage relations, one can cluster related cells based on inheritance of the same somatic mutations acquired by their common precursors. However, unlike single-cell transcriptomes which could be readily verified, the retrospectively built cell phylogeny is often untestable especially when the development involves stochastic decisions. It is therefore critical to know the methodology limits and further explore the optimum for cell phylogeny reconstruction via theoretical modeling.

The developmentally acquired somatic mutations that one can retrieve from extant cells guide the reconstruction of cell lineages and also limit the extent and resolution of the built cell lineages. Although the genome can potentially carry practically unlimited spontaneous mutations [1], mapping them thoroughly for every sampled cell is not only technically challenging but also financially unrealistic. By contrast, the CRISPR technology based on gRNA-directed mutagenesis by Cas9 allows generation of "genetic barcodes" consisting of well-defined units that one can assess with specific primers. Versatile designs are possible with such CRISPR-based genetic barcodes. First, the virtually infinite gRNA variants permit recruitment of diverse endogenous and/or exogenous sites into the barcode. Second, one gRNA target spans only 23 base pairs and a densely packed coding block can carry 40 targets per kb. Third, the CRISPR-induced mutations within a given gRNA target site or across adjacent sites are largely predictable and thus programmable. Fourth, there exist variant Cas9-related enzymes that can elicit distinct mutations via cutting the DNA or modifying specific nucleotides. Finally, one can restrict Cas9 activity to cycling cells to prevent post-mitotic mutagenesis. We can also drive Cas9 expression only in the lineages of interest for targeted cell lineage analysis. The enormous versatility plus its superior programmability and trackability make CRISPR as a uniquely powerful tool for genetic barcoding of cell lineages. Establishing a sophisticated system for modeling CRISPR-coded cell lineages would expedite the exploration of creative CRISPR barcode designs to map complex cell lineages ideally with single-cell-division resolution.

As in the phylogenetic analysis of biological species, the task for reconstructing cell lineage is to build 'trees' where leaves represent the examined objects (i.e. sampled cells) and the branching patterns outline the implicated developmental history. Because of these similarities, the phylogenetic methods are often employed for reconstructing cell lineage trees. The key elements in reconstruction are those internal nodes that represent the precursors giving rise to differentially marked descendants. The strategies for estimating those internal nodes therefore govern how to derive a tree that best describes the phylogenetic relationships of leaves. The underlying assumptions in deriving these strategies, however, are different between evolutional phylogeny and CRISPR-coded cell lineage tracking. For example, the number of sites of phylogenetic mutations are often regarded as infinite, whereas the number of CRISPR-coding target sites are usually very limited. The CRISPR-based mutations are often fixed once mutated (except for the one based on homing CRISPR [2]), whereas the evolutional mutations are transitional. Given these differences, CRISPR-based method requires independent investigation of optimal reconstruction methods and of intrinsic limitations of encoding process.

Knowing the general topology of underlying cell lineages is also critical for informative cell phylogeny studies. There are two basic modes of cell division that result in daughter cells with distinct (asymmetric) or equivalent (symmetric) potentials. Via asymmetric cell division, one stem cell can make an intermediate precursor with reduced potential and regenerate the stem cell. A stem cell can undergo multiple rounds of asymmetric cell division and deposit a series of post-mitotic cells in a linear manner (type 1). By contrast, a developing tissue may expand exponentially through symmetric cell divisions of existing cells (type 3). Distinct challenges exist in tracking protracted stem-cell-type lineages across many cell generations versus resolving the lineage relations among the exponentially increasing cells that potentially carry more coincident mutations. A versatile genetic system for cell phylogeny studies should work robustly for mapping both types of cell lineages, which needs pre-validation in silico. Here, we evaluated a multitude of algorithms for both types of cell lineage topologies.

Via theoretical modeling and computer simulation, we explored the design of CRISPR barcodes as well as reconstruction methods of CRISPR-coded cell lineages. First, we modeled the dynamics of the CRISPR-encoding process assuming constant Cas9 editing rate across targets and cells. This assumption leads to an exponential reduction in the number of new Cas9 edits along the lineage depth, indicating the exponential reduction in the capabilities in precise reconstruction of the lineages. In search for an optimal tree-building method, we evaluated the depth-dependent error rates in reconstruction algorithms previously employed for CRISPR-based lineage tracking as well as multitude of different variants of hierarchical clustering methods. Intriguingly, the most faithful tree building was achieved through hierarchical clustering with a distance metric based on the shared edits (similar to Russell-Rao metric) and complete linkage method. We found that the distance matrix derived from shared edits contains the information of the lineage depth in a way best suitable for reconstructing in a proper order.

Despite the best reconstruction method, the exponential reduction in available coding units severely constrains the trackable lineage depth. To overcome this limitation without drastically increasing the total number of encoding units, we further developed effective strategies for releasing available coding units gradually along the lineage progression. Encouragingly, use of parallel gRNA cascades or variable Cas9 editing rates could readily double or even triple the error-free depth range. Taken together, the CRISPR technology promises to facilitate dense reconstruction of cell lineages, essential for comprehensive analysis of organism development.

Results

Dynamics of CRISPR barcode editing across cell generations

To understand the dynamics of encoding process, we model the process of editingnU units of targets where the rate of editing per cell division is r.We assume the editing rate is equal and independent across units/cell division (Fig.1A). Unlike traditional phylogeny, once a unit is edited, it cannot be edited again. Hence homing CRISPR encoding is excluded from the current model. We also denote the number of editing outcome as nL (number of levels), which have equal probability ($p = 1/nL$) of being chosen at each editing event. Most of these assumptions are simplified. For example, level choice is likely to be biased and editing rate could vary between units. However, we began with this simplified model to delineate the basic dynamics of CRISPR-barcoding with the relatively easy mathematics.

Having a certain number of new edits at each cell cycle is critical for high-resolution reconstruction of a lineage. We call the number of cell divisions experienced by a cell since the activation of the encoding system

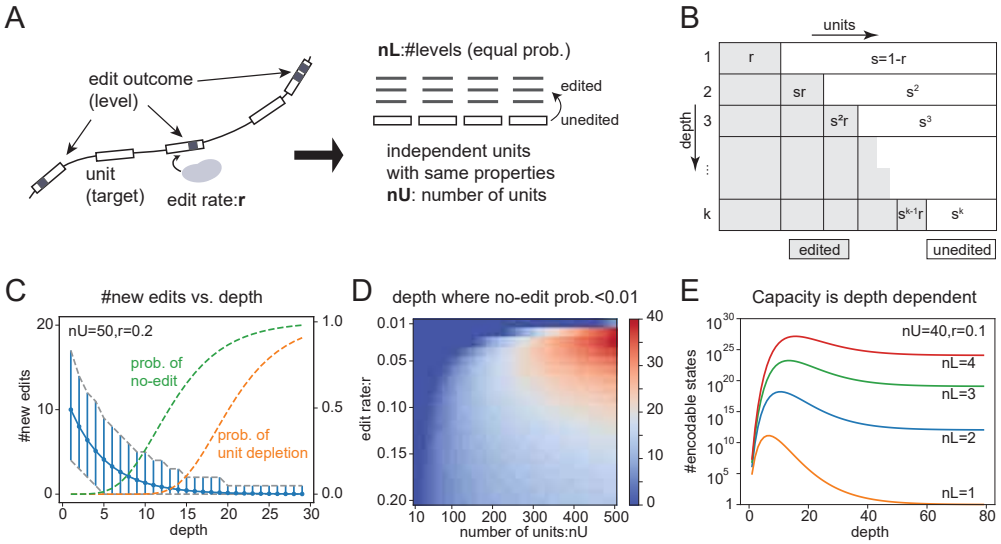

Figure 1: Dynamics of encoding

(A) Schematics of the model of the CRISPR encoding process. Critical prameters: nU: number of encoding units, nL: number of edit outcomes, r: edit rate/division. (B) Schematics showing how average numbers of edited units transition as cells divide. $s = 1 - r$. (C) Average ($nUrs^{k-1}$) and 99% range of new edits (calculated from binomial distribution $B(nU, rs^{k-1})$ at each cell cycle (blue lines), probability of no-edit (green, $(1 - rs^{k-1})^{nU}$ and unit depletion (orange, $(1 - s^k)^{nU}$), $nU = 50$ and $r = 0.2$. (D) Heatmap showing the maximum depth where probability of no-edit is smaller than 0.01 for ranges of edit rates (y-axis: 0.01 to 0.20) and number of units (x-axis: 10 to 500). ($z = argmax_k(1 - rs^{k-1})^{nU} < 0.01$, maximum depth where $P(noedit) < 0.01$, color-coded). (E) Capacity vs. depth for $nL = 1 \sim 4$. Capacity is the number of encodable states, and equals ${}_{nU}C_{ne} nL^{ne}$ where ne is the average number of edited units and at depth k :$ne(k) = nU(1 - s^k)$.

as depth. At given depth, under the above assumptions, the number of newly edited units follows a binomial distribution with the success probability as$r$ and the number of trials being the number of remaining unedited units. On average, the number of remaining unedited units decreases exponentially as $nU(1-r)^{k-1}$for depth of k; so does the average expected number of newly edited units drop exponentially as $nUr(1-r)^{k-1}$(Fig.1B, C). This phenomenon suggests that the ability to tell ancestor relationship in a lineage also decreases rapidly.

To record contiguous cell cycles along the lineage depth, , we need to know the distribution of the number of new edits at each depth. Since the convolution of binomial distribution is again binomial, this distribution can be easily calculated (see supplement) and the 99% range of the number of new edits at each depth is shown in Fig.1C vertical blue lines. One thing noticeable from this plot (other than exponential decay) is that the occurrence of "no-editing" (i.e. #new-edits=0) happens fairly before all the units are edited (Fig.1C green and orange lines). This indicates that encoding failure happens much earlier than depletion of unedited units. Fig.1D shows the largest depth where probability of no-edit is less than 0.01 for ranges of edit rates and number of total units. For a fixed edit rate, more units consistently sustain the encoding over a larger depth. For a fixed number of total units, there is an optimum edit rate. For example, for $nU = 20$, it requires edit rates of$0.3 \sim 0.7$ to achieve maximum encoding depth of two. By contrast, for $nU = 500$, edit rates of $0.02 \sim 0.03$ could support some new edits in each of 39 contiguous cell cycles. This analysis reveals that a fairly large number of coding units are required for dense reconstruction of protracted cell lineages.

We also need to ensure enough number of encodable states to confer co-derived cells with distinct codes, especially when mapping an exponentially increasing population of cells. Besides the number and possible assortment of cumulative edits, the number of levels per unit contributes significantly to the number of encodable states at a given depth. Fig.1E exemplifies how various numbers of levels (nL) may impact the average capacity (number of encodable states). First, this capacity is depth dependent since the number of edited units is depth dependent. Second, for $nL = 1$ the capacity converges to 1, because, all fully edited codes are identical. Therefore, for those systems with $nL = 1$, such as MEMOIR [3], a care must be taken to stop the encoding process while there is still enough encoding capacity. Third, when the number of units is reasonable, a small number of levels (except $nL = 1$) per unit can encode a fairly large system. Even with $nL = 2$, the coding capacity exceeds 10^{12} for $nU = 40$ and reaches 10^{150} for $nU = 500$. Therefore, except for covering an extremely huge system with limited units, the number of levels is less critical compared to the number of units or edit rate.

Most biological systems are relatively small (e.g. $< 10^{15}$ cells per human body) compared to the coding capacity of hundreds of units, but their production could involve rather protracted linear cell lineages (e.g. ~ 100 serial cell cycles made by a typical neural stem cell in the tiny *Drosophila* brain). Given the above modeling on editing dynamics, we should further explore how to maximize the depth of coverage with minimal number of coding units by possibly modifying the editing dynamics.

Robust and efficient encoding of protracted cell lineages

From modeling a simple scheme of cumulative editing (Fig.1C), we learn that having an average number of seven expected new edits can guarantee occurrence of at least one new edit for 99% chance to encode the ongoing cell cycle with 99% confidence. Using seven as a minimal number of new edits per cell cycle, we can figure out how many total coding units are needed for a given editing rate to record an entire lineage with 100 consecutive cell cycles. Notably, the optimal editing rate for recording 100 serial cell cycles is 0.01 with total required units of 1893 (Fig.2A middle). We define the efficiency of this optimal scenario as 37%, derived as the ratio of the total units minimally required to cover the entire lineage (700) over the actual required number of units. Lowering the editing rate reduces the efficiency because of leaving the majority of units unedited (Fig.2A left). By contrast, higher editing rates rapidly consume excessive units and entail astonishing numbers of total units to sustain some new editing throughout 100 serial cell cycles (Fig.2A right).

We calculate the coding efficiency across a range of lineage depth and a wide range of editing rates (Fig.2B). We learn that the maximal efficiency for recording serial cell cycles with simple cumulative editing can barely reach 40%. (In general, for desired depth of d, the optimum edit rate happens at $r = 1/d$ with efficiency of $(1 - 1/d)^{d-1}$, which is < 0.4 for $d > 6$.) Moreover, the coding efficiency is very sensitive to the editing rate which is often uncontrollable. For dense reconstruction of protracted cell lineages, we desire to establish more robust barcoding systems with higher efficiency by possibly modifying the editing dynamics.

To achieve a best possible coding efficiency with stochastic edits, we should maintain the average number of expected new edits around seven throughout the entire lineage. We envision two strategies to render the average number of expected new edits relatively stable across serial cell cycles, one controlling availability

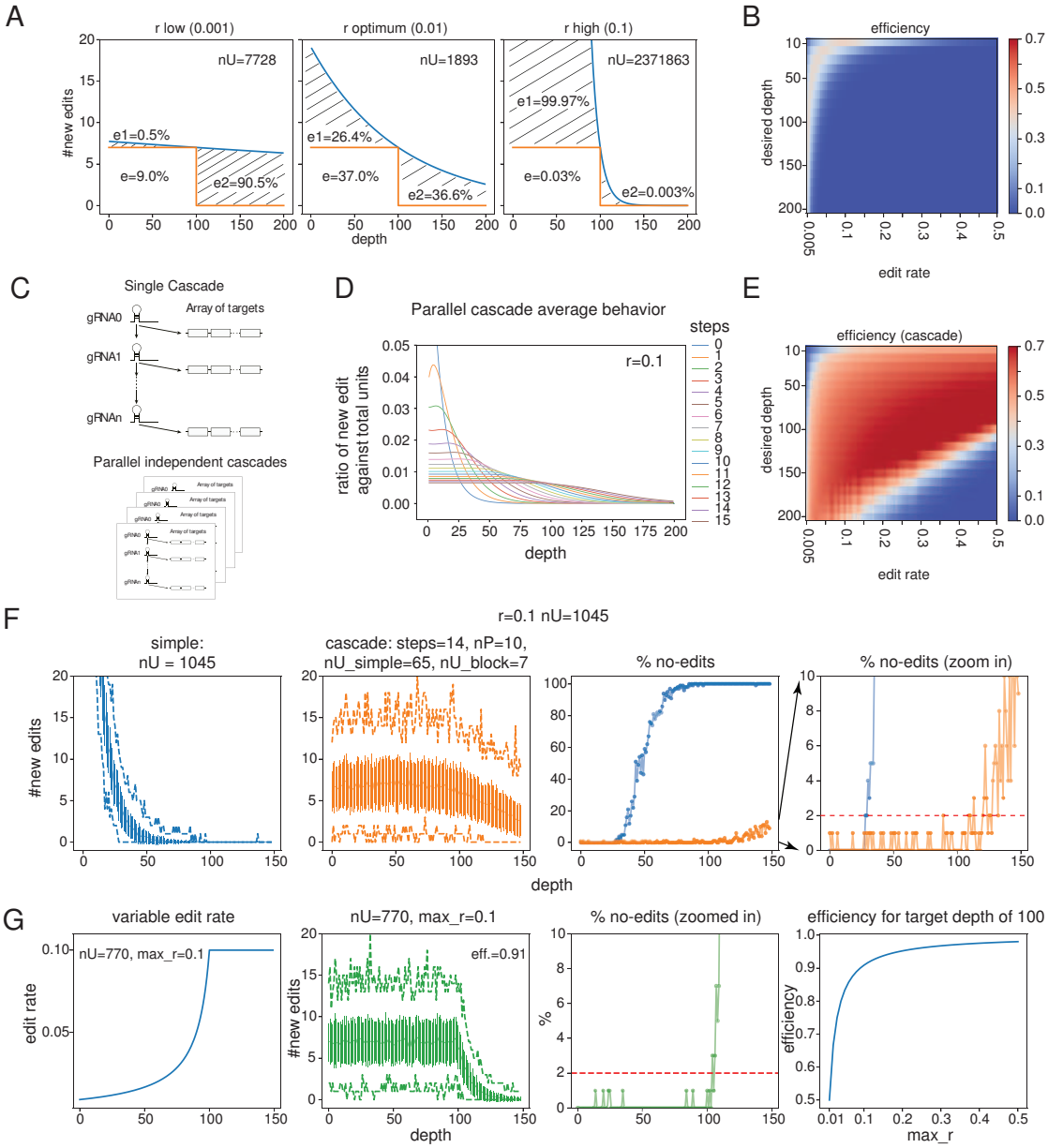

Figure 2: Dynamics of encoding

(A) Schematics showing the efficiency e for three cases of edit rates for simple exponential encoding process. The shadowed areas indicate wasted units. In the case of low edit rate (left panel), the waste resides mainly in the depth larger than the desired depth (100). In the case of high edit rate (right panel), the waste resides mainly in the initial phase. (B) Heatmap showing the efficiencies for various edit rates and desired depths for simple exponential encoding process. (C) Schematics showing a single cascade encoding process (top) and parallel cascades (bottom). (D) Average new edits (divided by total number of units) for parallel cascades with various number of cascade steps. Average is calculated as sum of infinite parallel cascades. (E) Similar to panel B, efficiencies of the cascade process for various edit rates and desired depths. (F) (Left two panels) Average number of new edits (middle solid lines), standard deviations (vertical bars) and minimum and maximum (dashed lines) of the number of new edits at each depth are shown for simple exponential case (1st panel) and for cascade process (2nd panel). Cascade process is realized with 10 parallel cascades of 14 steps with 7 target units per block and a single block of 65 pre-activated targets, constituting total of 1045 targets. (Righ two panels) Percentages of occurrences of no-edit at each depth are shown for simple (blue) and cascade (orange) processes. (G) (1st panel) An example of optimum variable edit rates when $max_r = 0.1$. (2nd panel) Similar to left two panels of F. Average, standard deviation, minimum and maximum of the new edits corresponding to the variable edit rate in the 1st panel are shown. (3rd panel) Zoomed in view of the percentage of no-edit. (4th panel) Optimum efficiency for different max_r in the case of desired depth of 100.

of units and the other controlling edit rate. The first one involves releasing various subsets of coding units for editing by distinct gRNAs in a step-wise manner. The second one depends on increasing the editing rate across serial cell cycles to compensate for the otherwise exponential reduction in the coding capacity along the lineage depth.

Supplying distinct gRNAs sequentially to edit separate pools of CRISPR targets chronologically should naturally lengthen the depth of coverage. The challenge is how to automate this process and distribute the overall coding capacity as evenly as possibly across an entire lineage. Recently, we have developed CaSSA [4], which permits gRNA-dependent gene reconstitution. Based on this system, we have further developed CLADES (Garcia-Marques et al., in preparation), which provides a universal strategy for gRNA-dependent gRNA reconstitution. This way, we can reconstitute distinct gRNAs in a preprogrammed sequence as a cascade. Like other CRISPR edits, the Cas9-dependent progression of the gRNA cascade is stochastic, so a single cascade produces rather uneven editings throughout the cell cycles. However, by combining multiple parallel cascades, the overall progression in the serial editing of distinct pools of CRISPR can be smoothened (Fig.2C, D).

For modeling such cascade-driven serial CRISPR edits, we assume comparable rates for gRNA reconstitutions and editing of discrete gRNA-specific targets. We further assume presence of infinite parallel cascades in the assessment of the best possible coding efficiency the cascade system can achieve. Plotted in Fig.2D are the average edits per cell generation (shown as percentage of total units) for various numbers of cascading steps. Intriguingly, as the cascading steps increase from 2 to 15, the distribution of edits across the depth becomes gradually flattened and the flattened portion can readily extend beyond 100 serial cell cycles. Utilizing such plots, we can derive the highest efficiency, possibly achieved with parallel gRNA cascades, for various combinations of editing rate and desired depth (Fig.2E). Compared to simple cumulative CRISPR edits, the cascade-based system has not only increased the maximal efficiency from 0.37 to 0.74 but also greatly broadened the high-efficiency domain across a wide range of editing rates (Fig.2B vs. Fig.2E). As to practical applications with limited parallel cascades, we demonstrate via computer simulation that a system with 10 parallel 14-step cascades to drive editing of 1045 units in total can effectively cover 100 serial cell cycles with ~ 7 new edits per cell cycle. Without cascades, the depth of coverage for 1045 coding units drops from 100 to ~ 30. Taken together, driving CRISPR edits with parallel gRNA cascades can improve the coding efficiency by delivering a rather constant number of new edits per cell cycle until some desired depth.

One can also possibly maintain a rather constant number of new edits across serial cell cycles by increasing the editing rate along the depth as the number of unedited units drops. Strikingly, we can achieve a coding efficiency of 0.91 by varying the editing rate inversely proportional to the depth ($r_k = ne_0/(nU - k * ne_0)$, where $ne_0$is the desired number of edits per generation, which is 7 here). In such a dynamic system, the average number of new edits per cycle is kept at ne_0. We further restrict r_k from exceeding some defined maximum editing rate (max_r), which is 0.1 in Fig.2G. The resulting curve of increase in the editing rate mimics the ascending expression levels of some temporal genes (e.g. Syp) in the cycling neural stem cells. We can possibly ramp up the editing rate in a similar curve by simply placing Cas9 expression under the control of the *Syp* promoter [5]. In summary, modeling the dynamics of cumulative CRISPR edits has informed us how to encode protracted cell lineages with robust and efficient barcodes.

Faithful reconstruction of densely encoded cell lineages

With a set of CRISPR encoded cells at hand, the next question is how best it can be decoded. Table 1 summarizes lineage reconstruction methods used in previous publications. Due to obvious similarity in the problem settings, algorithms from phylogeny such as maximum parsimony [11] and neighbor joining [12] were tried. In addition, custom graph-based algorithm (LINNAEUS [8]), or hierarchical clustering with some specific distance metric and linkage method were tried ([10, 3]). However, whether there are any differences between these, one of them is the best, or there is other algorithm better than all of these is not clear.

To explore this question, we set out to evaluate all of these algorithms. In addition, we also explored an extensive combinations of linkage methods and metrics for hierarchical clustering method.

As we saw in the previous section, the number of new edits decreases rapidly depending on depth, therefore we expect the reconstruction reliability is also depth dependent. The reconstruction error rate should be much smaller for the earlier depth. Several between tree distances such as Robinson-Foulds metric [13] or pair-wise shortest path difference [14] etc. are used previously to evaluate reconstruction performances of the whole tree. Here, we adapted Robinson-Foulds metric in a depth dependent manner for a much finer understanding of the reconstruction error (Fig.3A, see legend and method for details).

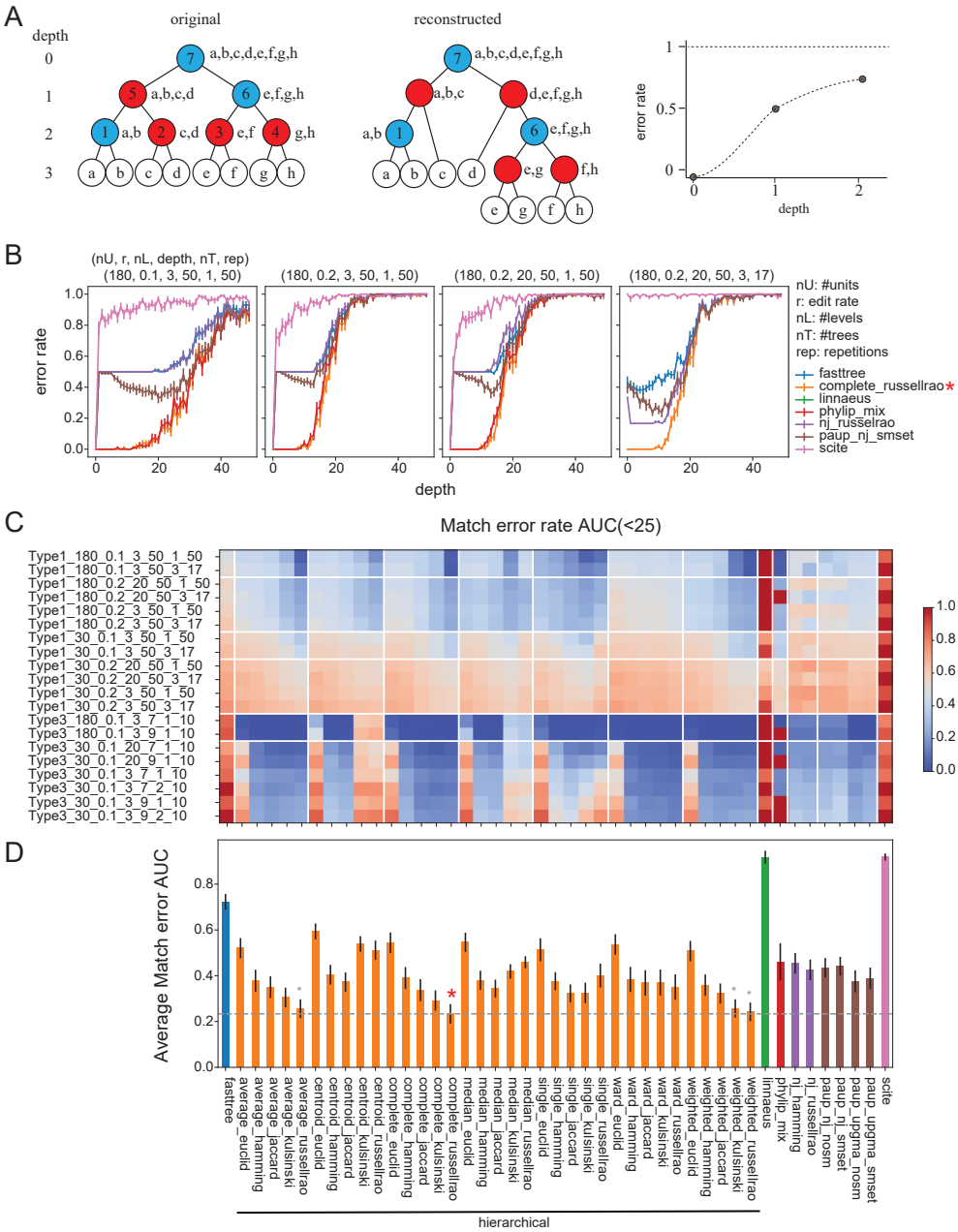

Figure 3: Performance evaluation of various reconstruction methods
(A) A simple example showing calculation of depth dependent error rate. The blue nodes in the original tree have matching nodes in the reconstructed tree but red nodes do not. At each depth in the original tree, the ratio of the number of the red nodes to the total number of nodes in that depth is the depth dependent error rate. (B) Depth dependent error rates for select conditions of inputs and reconstruction algorithms. The inputs shown here are Type1 (see method). Number of units (nU), edit rate (r), number of levels (nL) are as indicated. Number of trees (nT) indicate how many trees are combined and number of replicates (rep) indicate how many repeats are performed to generate the averaged traces. The errorbars indicate standard errors. Some conditions (such as linnaeus) does not show up which indicates that the execution took too long and terminated. (C) Heatmap of area under the curve (AUC) for depth <25 for all the conditions (inputs/algorithms) tested. The parameters for inputs are encoded as in panel B and shown in y-axis. The x-axis is common to next panel D and indicates the algorithms tested. For details on the inputs and algorithms see methods and supplements. (D) Averages of the AUC for all the input conditions. Error bar indicate standard error. For hierarchical clustering methods, linkage method (such as average, centroid) and distance metric (such as Euclid, hamming) combinations are indicated as (linkage)_(metric). Red star indicates the reconstruction method with the lowest error rate. Smaller gray stars indicate other methods statistically indistinguishable to the lowest case.

Assay Name	Tree Building Method	Comments	Ref
(sc)GESTALT	Maximum Parsimony by Camin- Sokal algorithm (using PHYLIP- mix, encoding each pattern as 0 or 1)	Since a pattern can span multiple targets, if each pattern is treated as a unit, then they may not be independent to each other.	[6, 7]
LINNAEUS	Custom algorithm (top-down, ordered by the abundance of scar among cells)	Uniqueness of scar, i.e. a same scar (pattern) does not happen as two independent events), is a critical assumption of the algorith, which is often not met.	[8]
MEMOIR	Hierarchical clustering with average linkage (UPGMA), with custom tie breaking. Also NJ(Neighbor Joining) and Maximum Parsimony are used for simulation.	MEMOIR "scratch pad" only has two states (intact and mutated). This only works for limited depth range, since if all the scratch pad are mutated, they cannot be distinguished.	[3]
Slavador-Martinez	Neighbor Joining using PAUP* and FastTree (for large samples)		[9]
Byungjin Hwang	LINNAEUS method as well as Hierarchical clustering with Jaccard distance metric.	Uses L1 repeats as targets and base editor for encoding. Largest number of units (>200)	[10]

Table 1: Summary of previously employed CRISPR-based lineage reconstruction methods

We simulated the CRISPR encoding process for a range of parameters (nU, r, nL) with different division modes (type 1: asymmetric division and type 3: symmetric division, see method for details). In some cases, 3 trees ($nT = 3$) are combined. From the simulated trees, leaf nodes are extracted as inputs. Each input is supplied to all of the surveyed algorithms and the reconstructions were evaluated against the original tree. For a same set of parameters, this is repeated 50 times (17 when number of trees is 3) and the average depth dependent errors were calculated. Plotted in Fig.3B are a subset of conditions showing the full average depth dependent error rates. As expected, earlier depths have smaller error rates. There are also clear performance differences between different algorithms. In some cases, (phylip_mix or linnaeus), the execution took more than 10 min and so terminated and flagged as failure. This is more than 1000 times the typical execution time of other algorithms (data not shown).

Fig.3C, D summarized the performances of different algorithms with various inputs. To simplify, area under the curve (AUC) of the depth dependent error rate (with $depth < 25$) is shown. For the failure case (execution time too long), this value is set to 1. From these surveys, we found that hierarchical clustering with complete linkage and russellrao metric performs best.

For the metrics for binary numbers, such as hamming distance, we extended their definitions to distinguish levels when $nL > 1$ (see supplement). Russellrao metric is one of the binary metrics and with this extension, it is 1 minus the proportion of the number of edited units which have the same level between two codes. Therefore, we call this "shared edits" (i.e. edits shared by two cells). With this phrase, we indicate the metric as well as the units which have the shared edits. Presumably these shared edits are inherited from a common ancestor. The complete linkage method takes the maximum element-wise distances of two clusters as the distance between these clusters. For shared edits metric, this is equivalent to assigning a code common to all the elements of the cluster to each cluster and then calculating shared edits distance of these newly assigned codes between clusters. The common code is equal to the original code except when a same unit is edited in two offsprings simultaneously and their edit outcomes (levels) happen to be the same. In summary, the hierarchical clustering with complete linkage using russellrao metric is the best performaning algorithm which also turns out to be a very natural way of reconstruction.

Why tracking shared edits outperforms more elaborate distance metrics?

In the previous section, we observed that the use of different metrics results in varied reconstruction performances in hierarchical clustering method. Hierarchical clustering iteratively merges a pair with the smallest distance and generate an ancestor node. Therefore, the choice of metric affects the choice of the pair being merged. To understand why shared edits metric outperforms other metrics, we first factored out a common

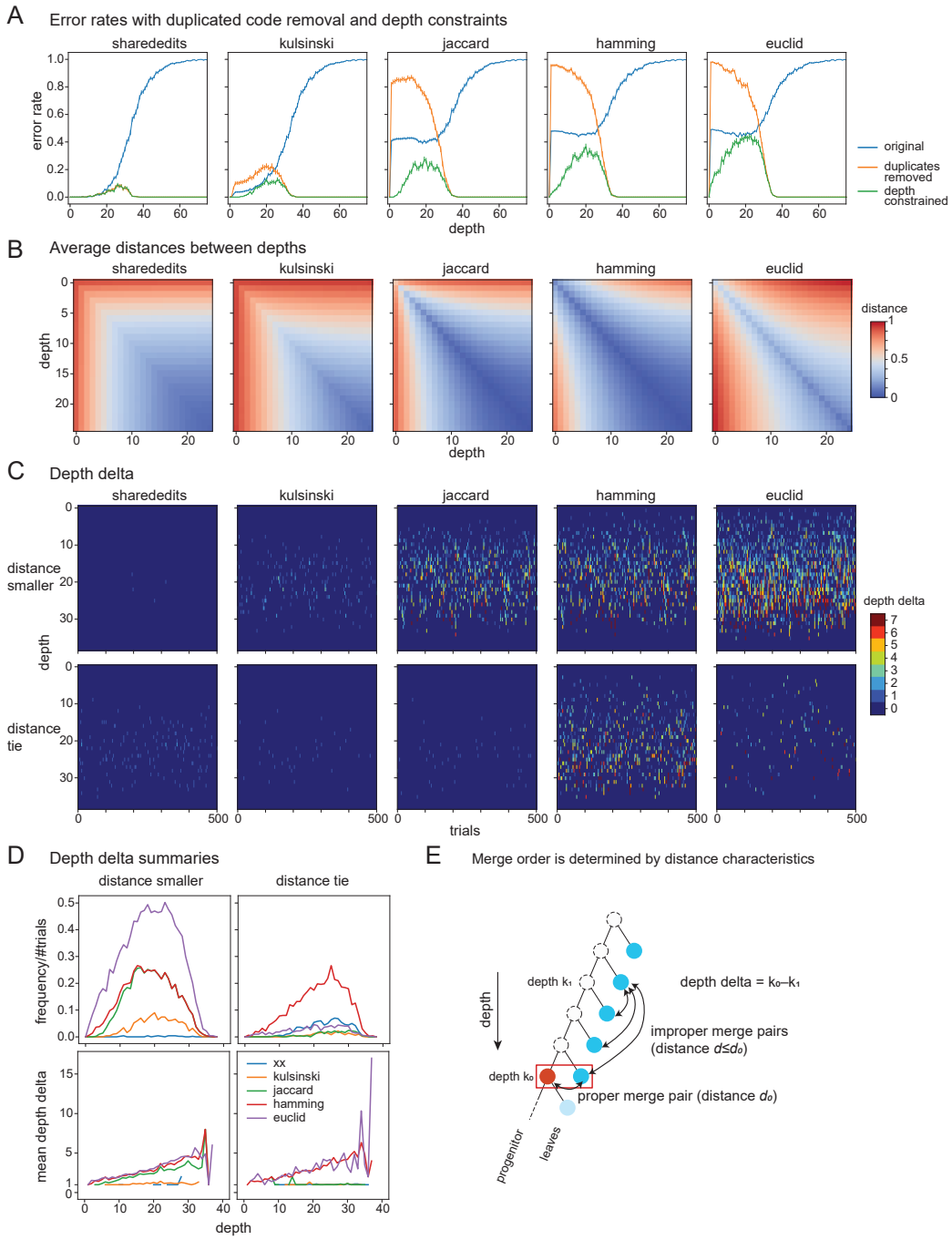

Figure 4: Why sharededits metric outperforms others

(A) Average depth dependent error rates for hierarchical clustering method with complete linkage for different distance metrics. Input is type 1, $nU = 180, nL = 3, r = 0.1, nT = 1$ and 500 replicates are used to calculate the average. Blue lines show the error rates for original inputs. Orange lines show the error rates after original inputs are modified to have no code duplication. Green lines show the error rates with depth constraint together with code duplication removal. (B) Pairwise average distances between nodes in different depths. Diagonals (k_0, k_0) are averages of distances between a progenitor and a leaf, (see panel E red and blue nodes at depth k_0). Off-diagonals (k_0, k_1) are averages of minimum distances between a leaf (blue node at depth k_1) and nodes in depth k_0 (both red and blue nodes at depth k_0). (C) Heatmap showing the color-coded "depth delta" (see panel E) for each depth (y-axis) and replicate (x-axis). (D) (top row) Average frequencies of distance swap or tie across trials. (bottom row) Average depth delta when distance is swapped or tied. (E) Definition of depth delta.

component, code duplication, which causes reconstruction errors regardless of the metric used. Code duplication between parent-offspring pairs occur when all the units are edited (unit depletion) or when remaining number of un-edited units is low which results in the absence of new edits (Fig.1C green line). Code duplicationcan also happen between non-parent-offspring pairs by chance but with much smaller frequency. Since no metric can distinguish cells with a same code, a code duplication results in a tie, which results in error. (In half of the cases if tie breaking is random.)

To remove the effect of code duplication, we modified the original trees to eliminate nodes with duplicated codes (see methods) and assessed the depth dependent error rate after reconstruction (Fig.4A orange lines). This procedure dramatically reduced the error rates for shared edit, kulsinski metrics but not for jaccard, hamming and euclid metrics. We investigated the places where errors are made and noticed that for the latter groups of metrics, there are a lot of precocious merging of shallower nodes. To see the effect of precocious merging, we then put a constraint that the depths of merging pair be less than 2 away from the bottom during reconstruction utilizing information from the original tree (Fig.4A green lines, see method). This constraint in fact drastically reduced the error rates for the latter three metrics. This indicate that the initial two metrics contain information of the depth suitable for proper reconstruction, whereas the latter three metrics do not.

To understand this in more detail, we plotted heatmaps showing the distances between nodes in different depths (Fig.4B). One noticeable difference between initial two and the latter three metrics is the diagonal part. In fact, shared edit and kulsinski metrics are not full distance metrics but pseudo metrics in that the self-distances are not necessarily zero. This turned out to be a prerequisite for a more relevant feature, that is, the distance of a pair of nodes is basically determined by the depth of the shallower node and monotonically decreases with depth (inverted L-shape patterns of the distance matrices). (To be precise, this is approximately so for kulsinski metric, as can be seen from the distortion of the inverted L-shape patterns in Fig.4B.) Because of this property, a precocious pair with shallower nodes is more likely to have a larger distance than the proper merge pair, thus preventing precocious merging of shallower nodes. Whereas with the latter three metrics, although the distances of the proper merge pairs (diagonal elements in Fig.4B) are on average minimum, due to the distortion surrounding the diagonal elements, we reasoned they can have more inappropriate precocious pairs having smaller or equal distance than the proper pair.

To see whether this is in fact the case, we detected the occurrences of distances of the pairs shallower than the proper merge pair being either smaller (swap) or equal (tie) to the distance of the proper merge pair (see Fig.4C, E). We separated the case of distance swap and tie because they contribute differently to the error rates. In the case of swaps, they always lead to errors, however, in the case of ties, the proper merge pair can be selected by chance. It turned out shared edits metric almost do not have distance swaps (Fig.4D top rows). kulsinski metric has less tie than shared edits but has much larger swaps. As expected, other three metrics have much higher occurrences of swaps (or ties). Moreover, when we quantified how much further away improper merge happens (depth delta, see Fig.4E), the latter three metrics have much larger values (Fig.4C, Fig.4D bottom rows), whereas shared edits and kulsinski depth delta is always less than 3 (Fig.4C, D). These indicate that overall distance characteristics of shared edits and kulsinski metrics (Fig.4B) prevented precocious merging of shallower nodes happening frequently and when it happens restricted it to a local range. Between, shared edits and kulsinski metrics, we found distance swaps happen more frequently in kulsinski and the depth deltas are larger (Fig.4D), thus shared edits performs better than kulsinski.

In summary, shared edit metric outperforms other metrics because it encodes depth information in a manner that enables the reconstruction to proceed in a proper order.

Discussion

CRISPR permits multiplex genome editing, which greatly facilitates engineering of intricate genetic barcodes for tracking cell lineages. Through theoretical modeling and computer simulation, we demonstrate the unparalleled promise of CRISPR-derived genetic barcodes in mapping protracted cell lineages with single-cell-division resolution. We learn that the patterns of shared edits across barcodes best depict cell lineage relations, a rather obvious observation which has been somehow overlooked in previous reconstructions of CRISPR-coded cell lineages. Appraising cell relatedness based on shared barcode edits permits the best reconstruction of cell phylogeny. However, the lineage depth one can track remains severely limited for standard encoding process especially when the total unit number is modest and significant editing rates are needed to ensure robust barcoding in earlier cell cycles. By utilizing parallel gRNA cascades enabled by CLADES (Garcia-Marques et al., in preparation) or using variable edit rates, we show it is able to slow down the

depletion of available coding units without sacrificing robust barcoding. With these encoding methods, we estimate that to densely track stem-cell-type lineages across 100 cell generations we need approximately one thousand independent coding units. Given CRISPR's enormous versatility, it should be possible to develop further sophisticated genetic barcodes for comprehensive analysis of whole-organism development.

Engineering CRISPR barcodes in vivo

Creating genetic barcodes for tracking cell lineages requires editing DNA sequences specifically during cell cycles. For continuous tracking, the earlier DNA edits should not be erased by later edits. Using CRISPR, we can derive genetic barcodes through editing pre-selected sites that may exist as clustered exogeneous gRNA targets [15, 8, 6] or distribute widely , like transposon-related repeats, in the genome [10]. As to tandem targets, one major concern is interference across nearby targets. Dual cuts by wild-type Cas9 can result in deletion of all intervening targets including those already edited [6]. Such complications would compromise overall coding capacity and could disrupt hierarchical clustering due to the loss of various earlier edits. Targeting endogenous repeats, by contrast, should better ensure independent edits. Encouragingly, one recent study has demonstrated the feasibility of editing endogenous L1 elements in tracking lineages of cultured cells [10]. However, involving endogenous sites requires detailed characterization of the genome to identify ideal targets (in terms of editing efficiency and edit retrievability) that play no role in normal development. Given the above pros and cons, one should also explore the possibility of dispersing exogenous gRNA targets (e.g. placing them sparsely on a bacterial artificial chromosome) to avoid inter-target deletions. Another solution is to create diverse barcodes via DNA base editing (e.g. use of Cas9-deaminase) rather than through repair of double-strand DNA breaks. In the absence of nuclease activity, one should be able to maintain independent edits among tandemly packed gRNA targets. However, one may need to increase the number of coding units drastically to compensate for the lower editing rates observed with existing Cas9 base editors.

The great versatility in the control of Cas9 further enhances the power of CRISPR in creating genetic barcodes. First, one should be able to modulate the editing rates by altering Cas9 expression levels. Moreover, one may possibly control Cas9 activity dynamically to achieve robust yet minimal editing in each cell cycle for efficient barcoding of protracted cell lineages. Second, one can express Cas9 in specific spatiotemporal patterns for tracking specific cell lineages at specific developmental stages. It is further important to restrict Cas9 to cycling cells to prevent post-mitotic editing, which are non-informative and may overwhelm precursor-derived edits especially in those long-lived extant cells (e.g. neurons). Finally, gRNAs govern the target specificity of Cas9 actions. The virtually infinite specificity of gRNAs endows an unlimited multiplex power, which can be exploited to build further sophisticated genetic barcodes (see next). Taken together, we see it feasible to record any dynamic biological activities across cell generations using CRISPR-derived genetic barcodes.

Robust and efficient encoding

Robust barcoding requires occurrence of some new edits in every cell cycle. The higher the editing rate is, the fewer coding units we need to ensure robust barcoding. However, the larger the editing rate (r), the faster the available coding units would drop due to an exponential reduction of the remaining units, thus preventing deeper encoding. This dilemma makes tracking protracted stem-cell-type lineages extremely challenging. Given a hundred of coding units, we need an editing rate of ~0.1 to achieve robust barcoding at the beginning. But having a hundred of coding units in total could not sustain robust barcoding at the rate of 0.1 beyond 10 cell generations (Fig.1D). To track 100 cell generations continuously as needed for mapping most *Drosophila* neuronal lineages, we would need a daunting number (>2 million, Fig.4A) of coding units if edited at the rate of 0.1. Notably, we can drastically reduce the needed number of coding units to around two thousand by decreasing the editing rate to ~0.01. However, still most of the units are not efficiently utilized (Fig.4A). To improve barcoding efficiency, we should instead modulate the availability of coding units or edit rates across the depth of cell lineages. For maximal efficiency, we should use up all coding units and distribute the edited units randomly and evenly throughout the entire length of each lineage.

Strikingly, using parallel gRNA cascades to drive the editing, we can consume most coding units and distribute the edits rather evenly throughout a protracted lineage. Reserving separate pools of coding units for editing at different developmental times has been practiced manually by injecting a second gRNA set at a later stage of zebrafish development [6]. Thanks to CLADES (Garcia-Marques et al., in preparation) which allows serial reconstitution of multiple gRNA variants, we can now automate the process of supplying distinct gRNAs in a cascade to edit different subsets of coding units serially. We found that infinite parallel

cascades mathematically produces a rather flat consumption of codes across depth (Fig.4D), enabling efficient encoding process compared to simple exponential case (Fig.4E). We also showed that this idealized behavior of infinite parallelism is essentially reproducible with rather small number (10) of parallel cascades (Fig.4F).

We also showed controlling edit rate can lead to very robust and efficient encoding system (Fig.4G). Even though the control of edit rate is approximate, the efficiency is expected to be very high. The edit rate may be controllable through driving Cas9/gRNA under endogenous genes which mimic the desired dynamics (such as *Syp* [5]) or under small molecule inducible promoters and manipulating the concentration of the inducer molecules. The exact configuration which may be able to achieve desired edit rate change is a subject of future studies.

Taking all factors into consideration, we propose a cascade barcoding system for tracking *Drosophila* brain cell lineages with ~1000 coding units in total. This elaborate system ensures occurrence of 7 or more new edits per cell cycle in 98% of time in all cell cycles, and consumption of >70% of the 1000 units by the end of 100 serial cell generations. Such cascade barcoding systems, driven by targeted Cas9 induction, would allow us to efficiently reconstruct cell phylogeny with single-cell-cycle resolution for any complex tissue.

A better cell lineage reconstruction method

Various reconstruction methods have been utilized for CRISPR-based cell lineage codes (Table 1). Here, we explored the performances of these methods as well as other numerous combinations of metrics and linkage methods in hierarchical clustering (Fig.3C). Notably, we found hierarchical clustering with previously unused metric outperforms all other methods (Fig.3D). This metric, which we call shared edits, is based on the number of common edits. If we assign to an internal node, a code common to all nodes below it, then the shared edits distance between internal nodes corresponds to complete linkage (since complete linkage takes the minimum of all the pairwise distances between the two groups of the leaf nodes below the internal nodes compared). Thus, the best method turns out to be a very natural reconstruction procedure.

We found that the reason shared edits metric outperforms other metrics is because it contains the depth information in a way appropriate for reconstructing the tree in a proper order (Fig.4). The shared edits distance of a pair is essentially determined by the number of edits the common ancestor of the pair has. This helps to prevent precocious merging of nodes in shallower levels and promotes a proper bottom-up reconstruction of cell lineages. Other metrics (Jaccard, Hamming and Euclid) do not possess this property and by imposing restriction in allowed depth during reconstruction, drastic ($> 50\%$) proportion of the reconstruction errors can be corrected (Fig.4A 3rd-5th panels, orange vs. green lines). Kulsinski metric has similar property to shared edits but depth constraint can still improve the error rate (Fig.4A 2nd panel) indicating the way it contains the depth information is still not ideal. In contrast, the shared edits metric error rates did not improve upon depth constraint (Fig.4A 1st panel) indicating the way shared edit contains the depth information is near ideal.

No reconstruction method can solve the ambiguities associated with code duplication. Thus, the errors caused by code duplication set the limit in the reconstruction error rates. By artificially removing code duplication, we found that the reconstruction using shared edits produces very small error rates (Fig.4A 1st panel orange line). This indicates the reconstruction method based on shared edits is very close to the optimum.

Since the residual errors (errors without code duplication) for shared edits metric is mainly due to distance tie (Fig.4C, D), we have tried to improve further by incorporating other metrics or factors (such as number of leaves under each node) and breaking the tie. These efforts produced very little to no improvements (data not shown). We have not yet tried to combine other methods such as Markov-chain-monte-carlo (MCMC) methods which incorporate scores from larger structures or later steps.

Additional needs for further sophisticated algorithms

For practical applications, we have further tested the robustness of our optimized tree-building method in multiple aspects, including variable editing rates and unequal choice of edit outcomes as well as substantial cell loss. We found it performing robustly as long as the barcoding is sufficient and intact. However, the program in its current state is extremely vulnerable to random loss of various coding units, which can happen in code retrieval process (e.g. single cell genomic PCR and sequencing). This weakness argues again for the importance of having integrated barcodes that can be retrieved readily as a whole which protects against random loss of codes. Nonetheless, there are clearly unmet needs, including recovery of missing codes and the scalability in processing big data sets, which we hope to address later with further sophisticated algorithms.

In summary, we have found a better method for reconstructing CRISPR-coded cell lineages than previously used. This method is based on the metric using the number of shared Cas9 edits. We have further demonstrated alternative encoding strategies in tracking every cell cycle across numerous cell generations with high efficiency. Such versatile genetic barcoding empowered by CRISPR technology will revolutionize how we study biological organisms.

Materials and Methods

Depth dependent error rate

There are several distance metrics that quantifies similarity between trees. Of these, Robinson-Foulds distance (RF distance) [13] is arguably the most frequently used. For two rooted trees, RF distance counts 0 for internal nodes which exist in both and 1 for which exist only in one tree (internal node match is determined by the set of leaves under the node). We extended this metric to depend on depth in the following way. We first designate one of the trees (usually the original simulated tree) as base tree and flag each internal nodes of the base tree depending on whether it has a matching internal node in the other tree (usually the reconstructed tree). Then for each depth of the base tree, we calculate the ratio of nodes which do not have a matching node within the depth (Fig.3A). This set of ratios is a natural extension of RF distance to depend on depth, albeit being normalized per depth. We call these ratios depth dependent error rates. There are other ways to extend whole tree metrics to depend on depth. For example, we can restrict the target of matching to be in the same depth in the other tree in the above example, or we can first restrict trees to subtrees down to a depth and then calculate whole tree distance between these subtrees as depth dependent distance. We have tried these metrics as well with similar qualitative results (data not shown). We have also extended another tree metric, *pairwise cell shortest-path distance* [14] in to a depth dependent metric using subtrees described above. This also resulted in qualitatively similar results (data not shown).

Lineage tree simulations

Here we tested two types of lineage trees. One increases linearly with depth (type1), and the other increases exponentially with depth (type3). A type 1 progenitor asymmetrically divides and generates one type1 and one ganglion mother cell (GMC) which turns into two terminal cells (leaf nodes). We simulated CRISPR encoding to happen once during a cell cycle before cell division, which resulted in the two offsprings of a GMC sharing one code, so we can simply say type1 progenitor generates another type1 progenitor node, one GMC and one leaf node. A type 3 progenitor symmetrically divides and generates two type 3 progenitors. At the desired depth, we stop the division and simply set those nodes without offspring to be leaf nodes. For the inputs to the reconstruction algorithms, only codes from leaf nodes are supplied.

Reconstruction algorithms

We tested reconstruction algorithms previously used in CRISPR-based lineage tracing publications. Of these Neighbor-Joining (NJ) [12] and maximum parisimony [11] are from traditional phylogeny field. We used the same software packages and settings as much as possible. In some cases, we also tested other implementation or settings as well. For NJ, we used PAUP* [16] with several different setting (see table below) and FastTree [17] as these are used in the previous CRISPR-based lineage tracing publications. We also tested *nj* function from skbio.tree Python package (http://scikit-bio.org) as an alternative implementation since it accepts input as distance matrix. For maximum parsimony, we used mix from PHYLIP package [18] which was used in GESTALT publications [6, 7]. We also tested a software package (scite [19]) from tumor lineage field as an example of algorithms which utilize a stochastic search algorithm such as MCMC method. For method used in LINNAEUS paper [8], we modified the author's R script to accept input in a more convenient format but used as is otherwise. For hierarchical clustering, we tested combinations of 7 linkage methods (average, centroid, complete, median, single, ward and weighted) and 5 distance metrics (euclid, hamming, jaccard, kulsinski, russellrao). Distance metrics are chosen to include popular ones (euclid, hamming) as well as ones used in previous publications (jaccard) and top performing metrics (russellrao and kulsinski) from preliminary survey of more extensive set of metrics. For binary metrics (hamming, jaccard, kulsinski and russellrao), we modified them to distinguish edit outcome levels (see supplement for exact definition). The *linkage* function in scipy.cluster.hierarchy Python package (http://www.scipy.org) is used to perform the hierarchical clustering. Most of the standalone programs output in nexus or newick formats. We used

DendroPy (http://dendropy.org) to parse these outputs. See supplemental material for exact details on algorithm used in Fig.3.

Removal of code duplication

To filter out the effect of code duplication, we first compared all the pairs of nodes using hamming distance and detected duplicated pairs as entries with 0. We then removed nodes with duplicated code with larger depth. When the node being removed is not a leaf node, child nodes of the node are reconnected to the parent node of the node being removed. In the case of trees used in Fig.4 (type 1, $nU = 180, nL = 3, r = 0.1, nT = 1$), this removal procedure shrunk the tree depth to < 40. The exact topology of the trees depend on the instance (trial).

Depth constraint in reconstruction

Depth constraint during reconstruction is enforced so that the minimum estimated depth of the next merging pair should be within the current maximum estimated depth minus 2. For leaf nodes, depths are assigned using the original tree information. For internal nodes with matching node in original tree, again depth information from the original tree is used. For internal nodes without matching node, depth is inferred as minimum depth of the nodes under the node minus 1.

References

[1] Dan Frumkin et al. "Genomic Variability within an Organism Exposes Its Cell Lineage Tree". In: *PLOS Computational Biology* 1.5 (Oct. 28, 2005), e50.

[2] Reza Kalhor et al. "A homing CRISPR mouse resource for barcoding and lineage tracing". In: *bioRxiv* (Mar. 12, 2018).

[3] Kirsten L. Frieda et al. "Synthetic recording and in situ readout of lineage information in single cells". In: *Nature* 541.7635 (Jan. 2017), pp. 107–111.

[4] Jorge Garcia-Marques et al. "Unlimited genetic switches for cell-type specific manipulation". In: *bioRxiv* (Nov. 14, 2018).

[5] Zhiyong Liu et al. "Opposing intrinsic temporal gradients guide neural stem cell production of varied neuronal fates". In: *Science* 350.6258 (Oct. 16, 2015), pp. 317–320.

[6] Aaron McKenna et al. "Whole-organism lineage tracing by combinatorial and cumulative genome editing". In: *Science* 353.6298 (July 29, 2016), aaf7907.

[7] Bushra Raj et al. "Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain". In: *Nature Biotechnology* 36.5 (May 2018), pp. 442–450.

[8] Bastiaan Spanjaard et al. "Simultaneous lineage tracing and cell-type identification using CRISPR-Cas9-induced genetic scars". In: *Nature Biotechnology* 36.5 (May 2018). 00033, pp. 469–473.

[9] Irepan Salvador-Martinez et al. "Is it possible to reconstruct an accurate cell lineage using CRISPR recorders?" In: *bioRxiv* (July 20, 2018).

[10] Byungjin Hwang et al. "Lineage tracing using a Cas9-deaminase barcoding system targeting endogenous L1 elements". In: *bioRxiv* (July 11, 2018).

[11] Joseph H. Camin and Robert R. Sokal. "A Method for Deducing Branching Sequences in Phylogeny". In: *Evolution* 19.3 (1965), pp. 311–326.

[12] N. Saitou and M. Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular Biology and Evolution* 4.4 (July 1, 1987), pp. 406–425.

[13] D. F. Robinson and L. R. Foulds. "Comparison of phylogenetic trees". In: *Mathematical Biosciences* 53.1 (Feb. 1, 1981), pp. 131–147.

[14] Edith M. Ross and Florian Markowetz. "OncoNEM: inferring tumor evolution from single-cell sequencing data". In: *Genome Biology* 17.1 (Apr. 15, 2016), p. 69.

[15] Anna Alemany et al. "Whole-organism clone tracing using single-cell sequencing". In: *Nature* 556.7699 (Apr. 2018), pp. 108–112.

[16] D.L. Swofford. *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4.* Version 4. Sinauer Associates, Sunderland, Massachusetts 2003. Sunderland, Massachusetts.

[17] Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin. "FastTree 2 - Approximately Maximum-Likelihood Trees for Large Alignments". In: *PLOS ONE* 5.3 (Mar. 10, 2010), e9490.

[18] J. Felsenstein. *PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.* University of Washington, Seattle., 2005.

[19] Katharina Jahn, Jack Kuipers, and Niko Beerenwinkel. "Tree inference for single-cell data". In: *Genome Biology* 17.1 (May 5, 2016), p. 86.

Supplementary Materials

Details of algorithm execution

Abbreviation in Fig.3	Package	Metric	Comments
fasttree	FastTree	n/a	Levels are encoded as character string (0~9,A~Z) and codes are then converted to FASTA format and supplied to standalone fasttree executable.
(linkage)_(distance)	scipy	euclid, hamming, jaccard, russellrao, kulsinski	Input to the scipy hierarchical clustering method is a distance matrix and which linkage method (average, centroid, complete, median, single, ward, weighted) to use.
linnaeus	LINNAEUS	n/a	Unit and level combination is treated as one entity (called scar) and its presence is indicated for each cell as an input. Linnaeus produces an output tree with scars as nodes. We converted this to a tree with cells as nodes by replacing scars with matching cells.
phylip_mix	PHYLIP	n/a	Unit and level combination is treated as one entity and its presence/absence is represented as 1 or 0. Thus, codes of length $n_u$are converted to strings of 0 and 1 of length $n_u n_l$. Weights are calculated as log ratio of counts of each unit/level combination to the maximum counts and encoded in a character array. Then mix program is executed with ifile,P,W,2,3,Y,wfile options, where ifile and wfile are the names of input and weight files.
nj_hamming	scikit-bio	hamming	skbio.tree.nj function takes distance matrix as an input, hamming distance is used.
nj_russellrao	scikit-bio	russellrao	Similar to above, russellrao distance is used.
paup_nj_nosm	PAUP*	n/a	Codes are converted as FASTA as in fasttree as inputs. NJ method is specified. No user stepmatrix is specified.
paup_nj_smset	PAUP*	n/a	As in above, but stepmatrix is specified with log of inverse frequency as the weights. (Same as in [9]).
paup_upgma_nosm	PAUP*	n/a	As in paup_nj_nosm but upgma method is used instead of nj.
paup_upgma_smset	PAUP*	n/a	As in paup_nj_smset but upgma method is used instead of nj.
scite	SCITE	n/a	Codes are converted to strings of 0,1 as in phylip_mix and supplied as an input to scite and executed with options "-i ifile -n nm -m nc -r 2 -l 3001 -fd 1e-10 -ad 1e-10 -cc 1e-10 -transpose", where ifile is input file name, nm is number of mutations, nc is number of cells

Table Supp.1: Settings for algorithms in Fig.3

Probabilistic Model of CRISPR Editing

The number of edited units, e_1, from total of n_u units with per cell division edit rate of r follows binomial distribution which we denote $B(n_u, r)$. That is, probability of getting e_1 edits is $P(e_1) = B(n_u, r)(e_1) = {}_{n_u}C_{e_1} r^{e_1}(1-r)^{n_u - e_1}$. The complement of this, i.e. the number of un-edited units, $n_1 = n_u - e_1$, follows $B(n_u, s)$, where $s = 1 - r$. In the second round of edits, getting e_2 new edits again follows binomial distribution: $B(n_1, r)$, but with a different parameter n_1 which depends on the outcome of the 1st round. The combined distribution is a multinomial distribution:

$$M(n_u; r, rs, s^2)(e_1, e_2, n_2) = \frac{n_u!}{e_1! e_2! n_2!} r^{e_1}(rs)^{e_2}(s^2)^{n_2}$$

Where $n_2 = n_u - e_1 - e_2$ is the number of unedited units after 2nd division. In general, after k cell division, probability of getting $e_1, e_2, \ldots, e_k$ edits at each division is a multinomial distribution:

$$M(n_u; r, rs, rs^2, \ldots, rs^{k-1}, s^k)(e_1, e_2, \ldots, e_k, n_k) = \frac{n_u!}{e_1! e_2! \ldots e_k! n_k!} r^{e_1}(rs)^{e_2} \ldots (rs^{k-1})^{e_k}(s^k)^{n_k}$$

Where $n_k = n_u - \sum_{i=1}^{k} e_k$ is the number of unedited units after k-th division (Fig.1B). Because of the property of the multinomial distribution (next section, Lemma 3), the total number of edited units after k cell division, $t_k = \sum_{i=1}^{k} e_k$, follows binomial distribution $B(n_u, r + rs + rs^2 + ... + rs^{k-1}) = B(n_u, 1 - s^k)$. Similarly, the number of unedited units n_k follows binomial distribution $B(n_u, s^k)$ with standard deviation $\sqrt{n_u s^k (1 - s^k)}$, (Fig.1C). We will define effective editing rate at depth k as $r_k = 1 - s^k$ and its compliment as $s_k = 1 - r_k = s^k$.

Properties of Multinomial Distribution

Lemma 1. *Sum of two random variables $x_{12} = x_1 + x_2$ in a multinomial distribution $M(n; r_1, r_2, \ldots, r_k)$ follows multinomial distribution $M(n; r_1 + r_2, r_3, \ldots, r_k)$.*

Proof. Using binomial expansion:

$$(r_1 + r_2)^{x_{12}} = \sum_{x_1 + x_2 = x_{12}} \frac{x_{12}!}{x_1! x_2!} r_1^{x_1} r_2^{x_2}$$

it follows probability for x_{12} which is a sum of x_1, x_2 over $x_1 + x_2 = x_{12}$ is:

$$\begin{aligned} P(x_1 + x_2 = x_{12}) &= \sum_{x_1 + x_2 = x_{12}} \frac{n!}{x_1! x_2! x_3! \ldots x_k!} r_1^{x_1} r_2^{x_2} r_3^{x_3} \ldots r_k^{x_k} \\ &= \left(\sum_{x_1 + x_2 = x_{12}} \frac{x_{12}!}{x_1! x_2!} r_1^{x_1} r_2^{x_2} \right) \left(\frac{n!}{x_{12}! x_3! \ldots x_k!} \right) r_3^{x_3} \ldots r_k^{x_k} \\ &= (r_1 + r_2)^{x_{12}} \left(\frac{n!}{x_{12}! x_3! \ldots x_k!} \right) r_3^{x_3} \ldots r_k^{x_k} \end{aligned}$$

thus, x_{12} (and $x_3, \ldots, x_k$) follows $M(n; r_1 + r_2, r_3, \ldots, r_k)$. □

Lemma 2. *Sum of random variables in multinomial distribution again follows multinomial distribution whose rate is the sum of rates of summed variables.*

Proof. Apply lemma 1 multiple times. □

Lemma 3. *Multinomial distribution integrated over all but one variable is a binomial distribution:*

$$\sum_{x_2, \ldots, x_k} M(n; r_1, \ldots, r_k) = B(n, r_1)$$

Proof. Applying lemma 2 on variables $x_2, \ldots, x_k$will yield the desired result since a multinomial distribution of degree 2 is a binomial distribution. □

Lemma 4. $\sum_{x=y}^{n} B(n, r_1)(x) B(x, r_2)(y) = B(n, r_1 r_2)(y)$

Proof. Using,

$$\begin{pmatrix} n \\ x \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \frac{n!}{x!(n-x)!}\frac{x!}{y!(x-y)!} = \frac{n!}{y!(n-y)!}\frac{(n-y)!}{(n-x)!(x-y)!} = \begin{pmatrix} n \\ y \end{pmatrix}\begin{pmatrix} n-y \\ n-x \end{pmatrix}$$

and replacing $n - x = x'$, where $x' : 0 \to n - y$, since $x : y \to n$,

$$\begin{aligned}
lhs &= \sum_{x=y}^{n} \binom{n}{x} r_1^x (1-r_1)^{n-x} \binom{x}{y} r_2^y (1-r_2)^{x-y} \\
&= \sum_{x'=0}^{n-y} \binom{n}{y}\binom{n-y}{x'} r_1^{n-x'} r_2^y (1-r_1)^{x'} (1-r_2)^{n-x'-y} \\
&= \binom{n}{y}(r_1 r_2)^y \sum_{x'=0}^{n-y} \binom{n-y}{x'} r_1^{n-x'-y} (1-r_1)^{x'} (1-r_2)^{n-x'-y} \\
&= \binom{n}{y}(r_1 r_2)^y \sum_{x'=0}^{n-y} \binom{n-y}{x'} (1-r_1)^{x'} (r_1(1-r_2))^{n-y-x'} \\
&= \binom{n}{y}(r_1 r_2)^y \{(1-r_1) + r_1(1-r_2)\}^{n-y} \\
&= \binom{n}{y}(r_1 r_2)^y (1-r_1 r_2)^{n-y} \\
&= rhs
\end{aligned}$$

□

Distance Metrics

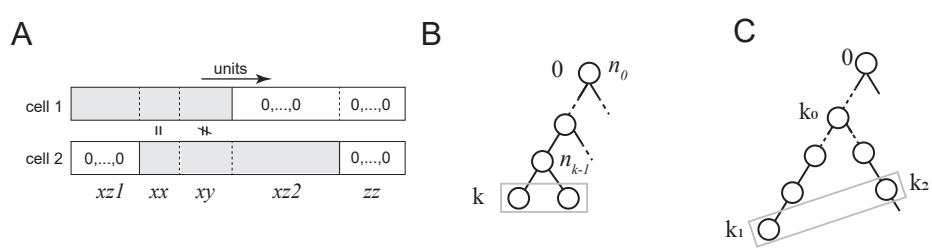

Figure Supp.1: 5 random variables for code match between two cells
The number of units which share the same edits between two cells is denoted xx. The number of units which are edited in both cells but not same is denoted xy. The numbers of units edited only in one cell are denoted $xz1$and $xz2$. The number of units unedited in both cells is denoted zz.

We denoted the number of outcomes of editing in a unit as n_l (excluding unedited state) and assumes the probability of choosing any outcome is the same, i.e.: $p = 1/nL$. When considering two lineage codes, there are 5 basic random variables which quantifies the difference between them (Fig.Supp1A). These are: X_{zz}, which is the number of unit unedited in both, X_{xz1} and X_{xz2}, which are the number of units edited in one but not the other, X_{xx}, which is the number of units edited in both and having the same edit outcome, and X_{xy}, which is the number of units edited in both but with different outcomes. For single round of edits, these variables sum up to the total number of units,$nU(= n_0)$, and follow multinomial distribution with rates rs (for X_{xz1} and X_{xz2}), r^2p (for X_{xx}), r^2q (for X_{xy}), and s^2(for X_{zz}), where $q = 1 - p$. When comparing two sibling codes, which are at the depth of k (Fig.Supp1B), we need to take into account the distribution of the number of unedited units at depth $k - 1$: n_{k-1}. For example, probability of getting X_{xy},is:

$$P(X_{xy}) = \sum_{n_{k-1}} P(X_{xy}|n_{k-1})P(n_{k-1})$$

$$
\begin{aligned}
&= \sum_{n_{k-1}} B(n_{k-1}, r^2 q)(X_{xy}) B(n_0, s_{k-1})(n_{k-1}) \\
&= B(n_0, r^2 q s_{k-1})(X_{xy})
\end{aligned}
$$

using Lemma 4 in the previous section. That is, rate is simply multiplied by s^{k-1}. Similarly for X_{xz1}, X_{xz2} and X_{zz}. For X_{xx}, we need to take into account shared code up to the previous depth $k-1$,which is $t_{k-1} \sim B(n_0, r_{k-1}) = B(n_0, 1-s^{k-1})$, so the rate becomes $1 - s^{k-1} + r^2 p s^{k-1}$. In general, if two codes are separated as in Fig.Supp1C, then, the rates for $X_{xx}, X_{xy}, X_{xz1}, X_{xz2}, X_{zz}$ are

$$
\begin{aligned}
X_{xx} &: r_{k_0} + s_{k_0} r_{k_{10}} r_{k_{20}} p = (1 - s^{k_0}) + s^{k_0}(1 - s^{k_{10}})(1 - s^{k_{20}}) p \\
X_{xy} &: s_{k_0} r_{k_{10}} r_{k_{20}} q = s^{k_0}(1 - s^{k_{10}})(1 - s^{k_{20}}) q \\
X_{xz1} &: s_{k_0} s_{k_{10}} r_{k_{20}} = s^{k_1}(1 - s^{k_{20}}) \\
X_{xz2} &: s_{k_0} s_{k_{20}} r_{k_{10}} = s^{k_2}(1 - s^{k_{10}}) \\
X_{zz} &: s_{k_0} s_{k_{20}} s_{k_{10}} = s^{k_1 + k_2 - k_0}
\end{aligned}
$$

where $k_{10} = k_1 - k_0$ and $k_{20} = k_2 - k_0$.

Using these variables, we can also define the (extended) distance metrics between two lineage codes as follows:

$$
\begin{aligned}
d_{russellrao} &= 1 - \frac{X_{xx}}{nU} \\
d_{kulsinski} &= \frac{X_{xz1} + X_{xz2} - X_{xx} + nU}{X_{xz1} + X_{xz2} + nU} \\
d_{jaccard} &= \frac{X_{xz1} + X_{xz2} + X_{xy}}{X_{xz1} + X_{xz2} + X_{xy} + X_{xx}} \\
d_{hamming} &= \frac{X_{xz!} + X_{xz2} + X_{xy}}{nU}
\end{aligned}
$$