Internal Report CS Bioinformatics Track 16-02 October 2016

# Leiden University

# Computer Science

# Bioinformatics Track

# Tracing scar-lineages and collapsing read error

Buys A. de Barbanson

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

# Tracing scar-lineages and collapsing read errors

**Thesis project report, 2015-2016**
Buys de Barbanson

**Supervisor:** Jeroen de Ridder (Tu-Delft/UMCU)
**External supervisors:**
Bastiaan Spanjaard (Hubrecht Institute)
Alexander van Oudenaarden (Hubrecht Institute)

**Leiden University supervisor:**
Fons Verbeek

# Contents

## Lineage tracing introduction

Eukaryotic organisms generally grow by dividing their cells in a binary fashion. In the cell division process a single cell duplicates its genome and splits into two daughter cells. Upon completion of the duplication, the ancestor cell has given rise to two cells which are genetically identical, apart from mutations which originate from errors introduced by copying the genome. Together, the millions, billions or sometimes even trillions of cells form a living organism consisting of multiple specialized cell types. Even though the molecular machineries and regulatory mechanisms of cell division itself are rather well understood, it is much less clear how the thousands of cell-divisions that take place in the developing organism are controlled in order to produce a complex eukaryote.

The cell divisions which lead to the organism at a specific point in time can be represented in a tree, which reflecting the binary fashion of mitotic cell-divisions is mostly binary. In contrast to phylogenetics where the edge weights are used to indicate time, edges in the cell-lineage tree do not have weights and solely describe the hierarchy of the observed ancestral relationships. Reverse engineering of an organism's lineage tree structure can be achieved by making use of heritable genomic markers. When a heritable marker is introduced in a cell, all descendants will carry this marker. If introduced early in the organism's development, when it consists of only a few 'founder' cells, these markers will indicate the ancestry to these founder cells. The leaves of the cell-lineage tree indicate cells present in the organism at time of sampling. Measuring the lineage markers in current cells of the organism teaches us if one or more of the marked founder cells give rise to certain parts of the organism, such a certain organ or limb or whether specialization is determined even earlier.

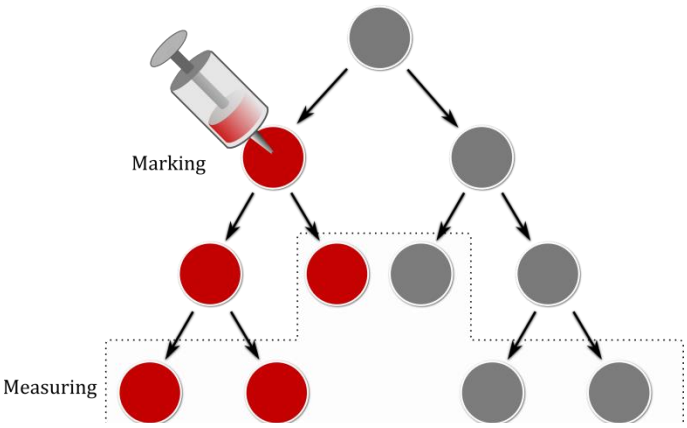



**Figure 1: Basic principle of lineage tracing. An organism starts as a single cell, and grows by cell-division. By introduction of a lineage-marker, ancestral relations between cells can be shown.**

In the early effort, cells were marked using dyes unable to exit the host cell and selected not to be degraded quickly. Dyes have the drawback of dilution over time and cell-divisions. Secondly the amount of dyes used limits the amount of cell-lineages which can be tracked in parallel. An historically important feat using this method is the complete cell-lineage tree of the *C. elegans*, which was mapped out in 1983[1].

Current state of the art methods use genomic markers which use the genome of the cell to store lineage information. Markers which are incorporated in the genome are useful as all genetic information of a cell is passed to its daughter cells making the marker persistent over cell divisions, and thus an ideal lineage marker. Secondly a genomic marker is useful as the range of possible markers is generally much higher than the markers used in conventional methods, allowing for tracking many lineages at the same time. When combined together with other measurements such as transcriptome sequencing, lineage tracing promises to help in revealing the processes which regulate cell development and drive the formation of the different cellular structures and ultimately the complete organism.

In cancer research often mutations are used as cell-lineage marker. As mutations are heritable all successive cells will carry the lineage marker. The drawback of this is that it is expensive to capture the mutations as they are scattered around the genome at random locations. Secondly it is impossible to change the mutation rates without affecting the host in a negative way. Thirdly, the baseline mutation rate of a healthy organism is lower than the rates observed in cancer, making the methodology due to a lower resolution less suitable for other applications. The main advantage of mutations as lineage marker is that due to the size of the genome, there are *3*N* possible lineage markers. The enormous amount of possible mutations results in a very low probability of the same marker being introduced multiple times.

An old lineage-tracing method becoming more popular due to the advancements of sequencing technology is the use of lentivirusses which incorporate a unique lineage-marker in the host genome[2]. As the viruses are designed such that they will be unable to exit the cell, a single virus particle will only introduce a single lineage marker. By using many viruses carrying a different bar-code, many lineages can be distinguished in parallel. As the introduced barcodes are flanked by known regions, the lineage markers can be selectively amplified using PCR.

## Scartrace method

An experimental high throughput lineage tracing method is being developed at the Hubrecht institute: scartrace[3]. In this method an organism with multiple identical synthetic sites encoded in the genome of every cell is used. In the single cell stage all sites are unaltered, we will identify this site-state as being wild-type (*Wt*). As the organism grows by cell division, a cutting enzyme (Cas-9) which is targeted to cut the unaltered sites is gradually introduced. The enzyme cut and the innate imperfect DNA repair mechanisms change the state of the targeted site permanently. The altered state of the site is identified as “scarred” referring to damage which has been repaired while leaving permanent traces. Because the synthetic sites are located on a (single) chromosome, a copy of all sites in their respective state is inherited by both descending cells upon cell division.

There are a couple of advantages of the scartrace method over using mutations. The coverage for the Hubrecht scar tracing method is high in comparison to coverages common in whole-exome sequencing of a tumor. The result of the high coverage is that relative differences in the frequencies of scars will be easier to pick-up. Secondly the scarring rate is much higher than the baseline mutation-rate. In cancer the mutation rate is often higher than the baseline mutation rate as in cancer the repair mechanisms are down-regulated, making mutations a less useful lineage marker in non-cancerous situations[4]. Compared to the use of lentivirusses to induce the markers, scartrace is less disruptive for the host. Lastly the method is relatively accessible as many organisms with GFP-transgene integrations are available. Using eGFP as target allows confirming whether scarring has occurred, the loss of fluorescence indicates the formation of scars[3].

The scartrace experiments starts in the single cell stage, then organism will be scarred gradually over time. This gradual scarring process incorporates lineage information into the scar-sites in the organism. This information can be retrieved by amplicon-sequencing the DNA or RNA (when the scar site is located in an expressed gene) of cells of interest. Using the lineage information obtained by sequencing, information of the lineage tree can be inferred using algorithms which perform lineage inference, with approaches which are similar to phylogenetic tree inference.

## Outline

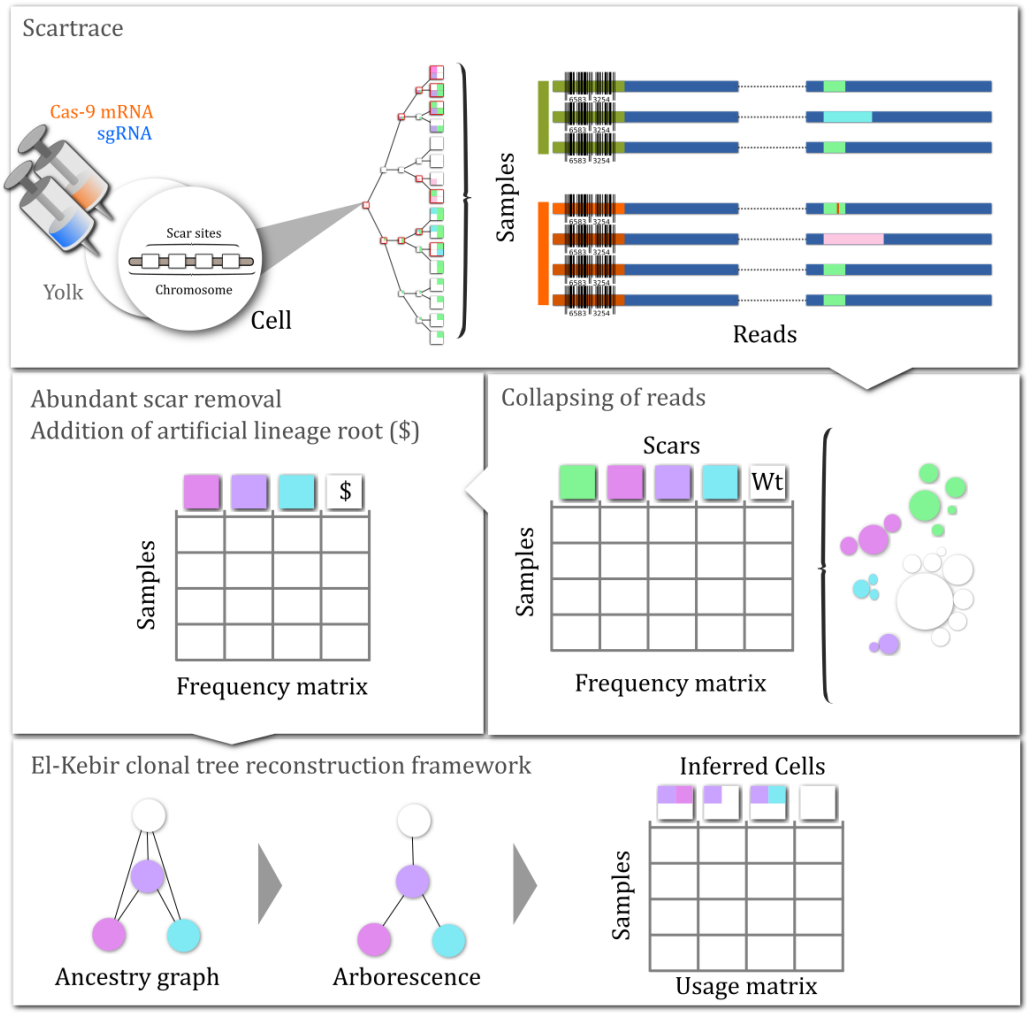



**Figure 2: Overview of methods used. The scartrace protocol induces genetic markers in an organism. Samples from this organism are dissected and the scars are measured using amplicon sequencing. The resulting reads contain errors which are removed by collapsing the reads. Scars which are likely to be introduced multiple times are filtered from the dataset. A lineage root frequency column is added to the matrix and the Wildtype (Wt) frequencies are removed.**

The research question addressed in this thesis is whether it is possible to obtain lineage information from samples in which scars are induced by the effect of a Cas-9 induced DSB and non-homologous DNA repair. The introduced scars function as lineage markers and are measured by amplicon sequencing. Sequencing errors make it challenging to identify which markers are present in a mixed sample. To tackle this pre-processing problem we design an amplicon-read collapsing method which uses the base-calling confidence values and sequence abundances to perform collapsing, and gives visual feedback of the original dataset and the results. This helps in determining which, and how many markers are present.

The next challenge is how to extract lineage information from the samples. As the use of scar-lineage markers in bulk samples is new, there are no directly applicable analysis available. The computationally difference to using somatic mutations as lineage marker is that while the exact same somatic mutation is very unlikely to happen twice, some scars occur many times throughout time. To transform the scartrace data into data which is compatible with existing frameworks, only the scars

which are likely to be introduced a single time are taken into account. A proof-of-concept lineage tracing algorithm is created by building upon an existing cancer-based lineage tracing algorithm.

## Sequencing errors

When using genomic markers, sequencing is used to estimate the marker presence and abundance. The drawback of using sequencing is the introduction of sequencing noise, which mostly results in many false positive marker counts. In case of scartrace, the amount of unique sequences present in the reads from a sample is much higher than the amount of distinct scars truly present in the sample. In practice it is challenging to filter these reads as the expected amount of markers is unknown, secondly the base-calling quality varies between sequencing runs. Quality control thresholds have to be configured per run, and appropriate parameters are challenging to estimate because there is no straightforward way to assess the effect of the filtering used. In order to perform proper lineage tracing analysis the errors induced by sequencing have to be dealt with. Ultimately resulting in datasets where the amount and abundance of markers is corrected, by collapsing most of the noise introduced by sequencing.

The original scartrace and other similar methods use alignment to a reference sequence to describe lineage markers in an effort to mitigate the effect of sequencing noise[3,5]. The effect of noise is reduced as the alignment of a read is less affected by substitution errors, the most common type of errors for the Illumina platform. By describing the lineage markers by their alignment, substitutions in respect to the reference are considered the same lineage marker. The problem with using alignment to identify correct scars is that it is possible that two scars share the same alignment to the reference. Secondly it can be inconvenient to enforce the sequence of the scar-site. When the exact scar-site sequence(s) are unknown, then alignment is unfavorable.

### Visualizing and correcting lineage marker datasets

Given the knowledge about the errors which are induced by the Illumina sequencing platform[6–8] it is known that a biological sequence will produce reads which exactly match the biological sequence, but also sequences which have one or more substitutions. The Hamming distance between the real biological and erroneous sequences will thus be low. Deletion errors, where bases in the biological sequence are skipped and insertion errors, where random bases are inserted into the read are uncommon. Often these deletion and insertion (indel) errors will induce a large Hamming distance to the biological sequence. A local alignment results in a score which can be used to estimate the amount of edits between a sequence containing indels and the real biological sequence.

Sequencing a single set of identical molecules reveals the extent to which erroneous reads are produced. For such a dataset the distribution of edit distances between sequenced reads and the real biological sequence are shown in figure 2. The majority of the reads is equal to the expected biological sequence. The set of reads containing substitution error is quite substantial, about 12% of the total abundance of the biological molecule yields a read containing a substitution error. For a small fraction of the reads, indel errors are observed. Less than 0.5% of the reads are very distant from the biological sequence. In this particular dataset one connected component in the graph contains a sequence from another sample. The remaining artifact reads contain homopolymers, which mostly consist of long stretches of cytosine base calls (see supplement).

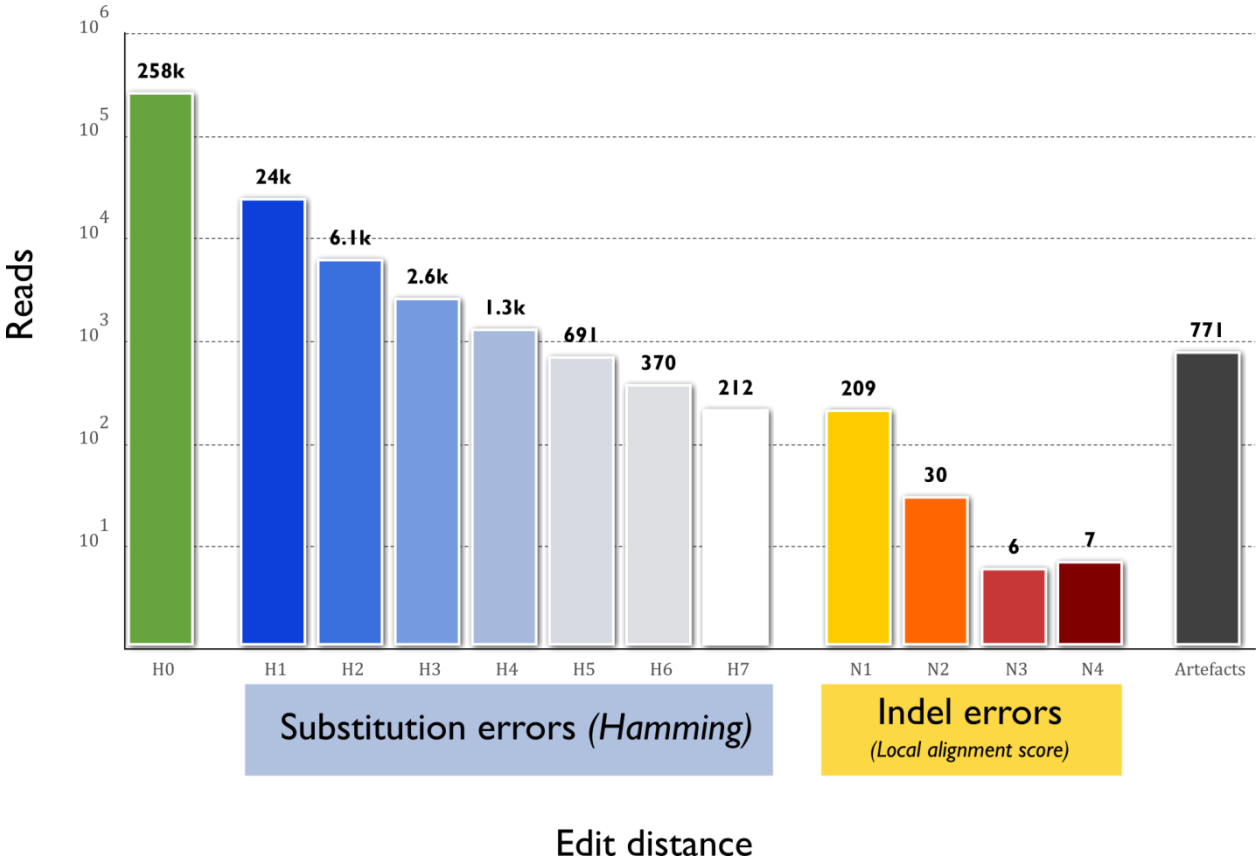



**Figure 3: A single sequence was sequenced, reads exactly matching this sequence are shown in green. The erroneous reads can be seen at hamming distances > 0. Sample shown is a DNA amplicon sequencing dataset of a Cas-9 protein injected fish without injecting a synthetic guide RNA (sgRNA). The sgRNA is used to target the scar sites, without sgRNA the Cas-9 will not cut the scar-sites and we expect to sequence only unaltered scar-sites. The local alignment score is calculated by (local) Smith-Waterman alignment to the expected sequence. The alignment is parameterized to have all costs (substitution/deletion/insertion) set to one. Sequences having a distance higher than 4 edits and a Hamming distance higher than 7 from the expected sequence are considered to be an artifact. All edit distances shown are calculated by alignment and are not based on approximations.**

The Hamming distance metric can be used to cluster sets of reads together which might originate from the same biological sequence. A way to represent the relation between the reads in a sample is by using a graph. The Hamming-distance relation between all reads of a single sample can be estimated by calculating a pairwise hamming distance matrix. A sparse variant of this matrix, which only contains distances equal to one, is used to construct an undirected graph. In this Hamming graph, each node corresponds to a unique sequence. The edges in the graph indicate a Hamming distance of 1, which corresponds to a difference of only a single base-call.

By using a force-directed layout algorithm, Hamming graphs allow for intuitive visualization of amplicon datasets. Biological sequences show up as central hubs surrounded by nodes which contain sequences with substitution errors. Coloring the nodes by the average-base calling confidence adds additional guidance to the visualization by using the property that most erroneous sequences have a lower average-base calling confidence compared to the completely correctly base-called biological sequence. An example of a Hamming graph for a single biological sequence is shown in Figure 4 and Figure 5.

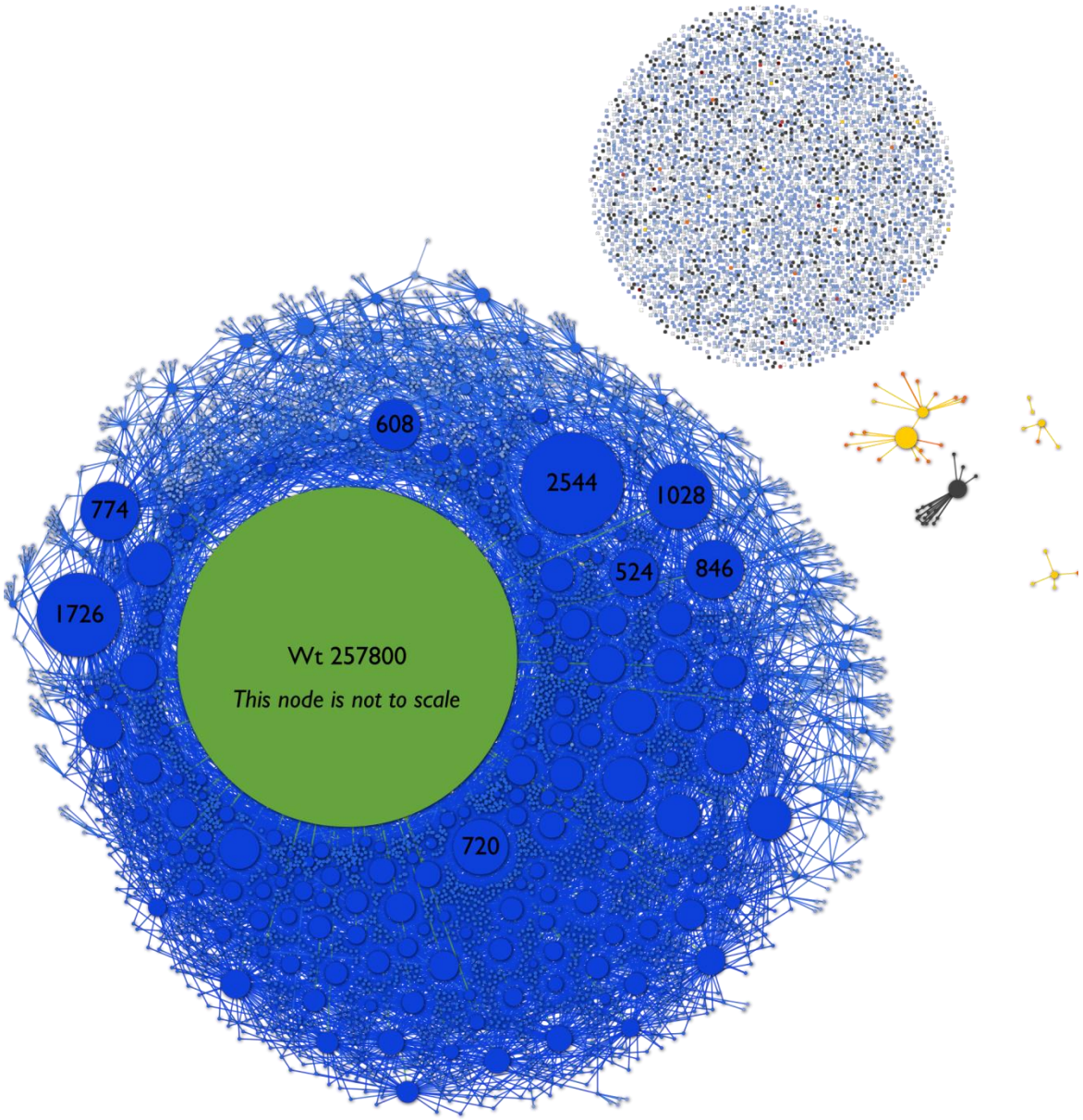



**Figure 4: An approximation of the Hamming graph for a sample where only a single biological sequence was sequenced. The node colors used match the colors in the corresponding edit-distance histogram in Figure 3. Edges indicate a Hamming distance of 1 between two sequences. Unconnected nodes are positioned in the top right circle. Errors are not distributed evenly, some errors occur more than others, resulting in a variety of sizes in the nodes around the *Wt* sequence. Apart from the unconnected sequences in the top right, the separate components in the Hamming graph are caused by indels. The H1 connectivity already enforces a very tightly connect graph, not many nodes are completely unconnected. The grey connected component is suspected to be a sample bleed-through effect. The graph layout was generated in Gephi and the Python scripting plugin. Graph graphics generated using a custom SVG generator and rendered using InkScape.**

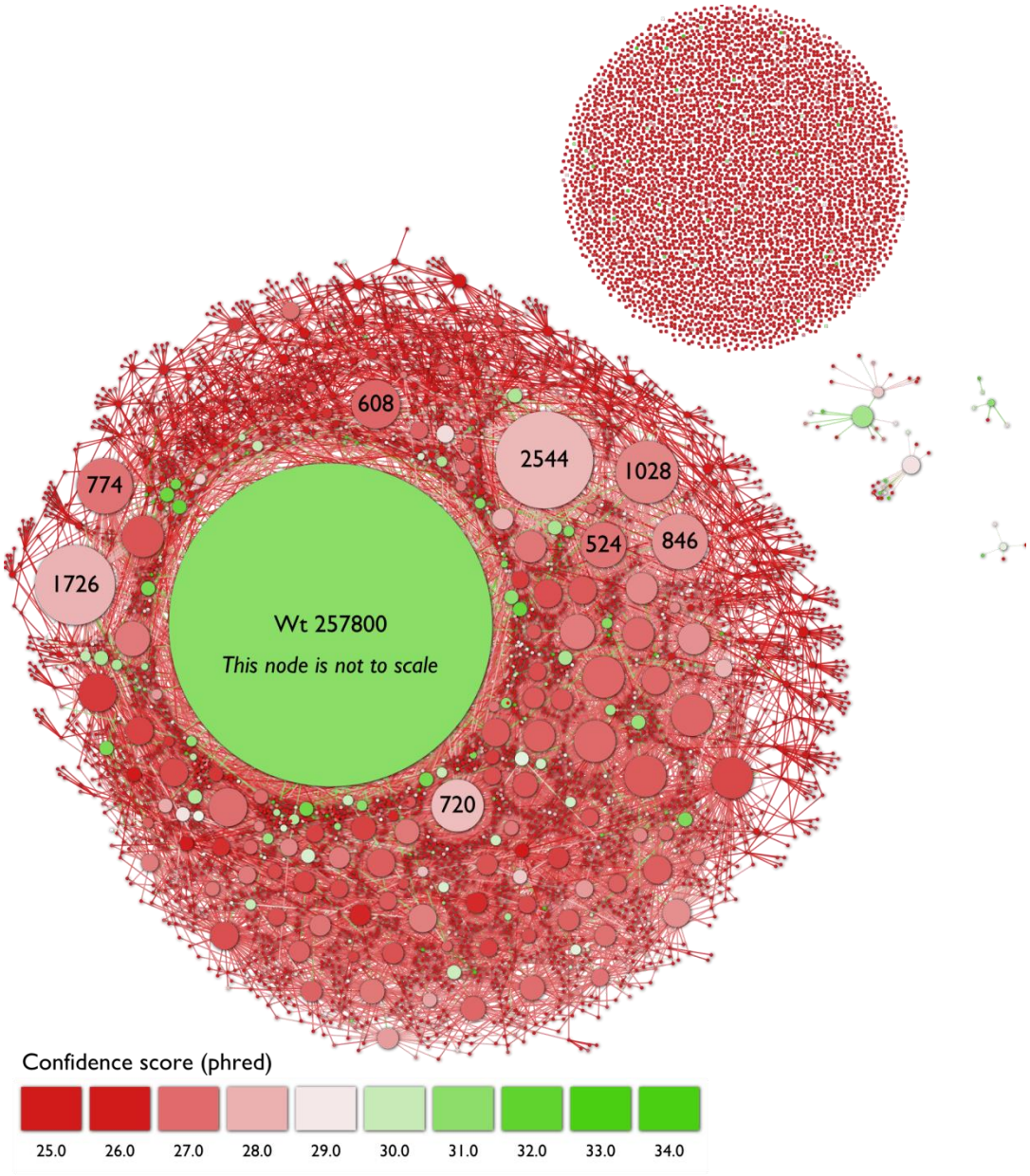



**Figure 5: Hamming graph for a single biological sequence. Nodes indicate unique sequences, edges indicate a Hamming distance of 1.**

Given enough sequencing depth, a biological sequence and the majority of its respective erroneous reads will be member of the same connected component. The sum of the correct and erroneous reads member of the same connected component approximates the amount of times the biological sequence was sequenced. The abundance of the complete component reflects the abundance of molecules with an identical sequence. Taking also the close-erroneous reads better reflects the amount of molecules present than only the abundance of the exact sequence.

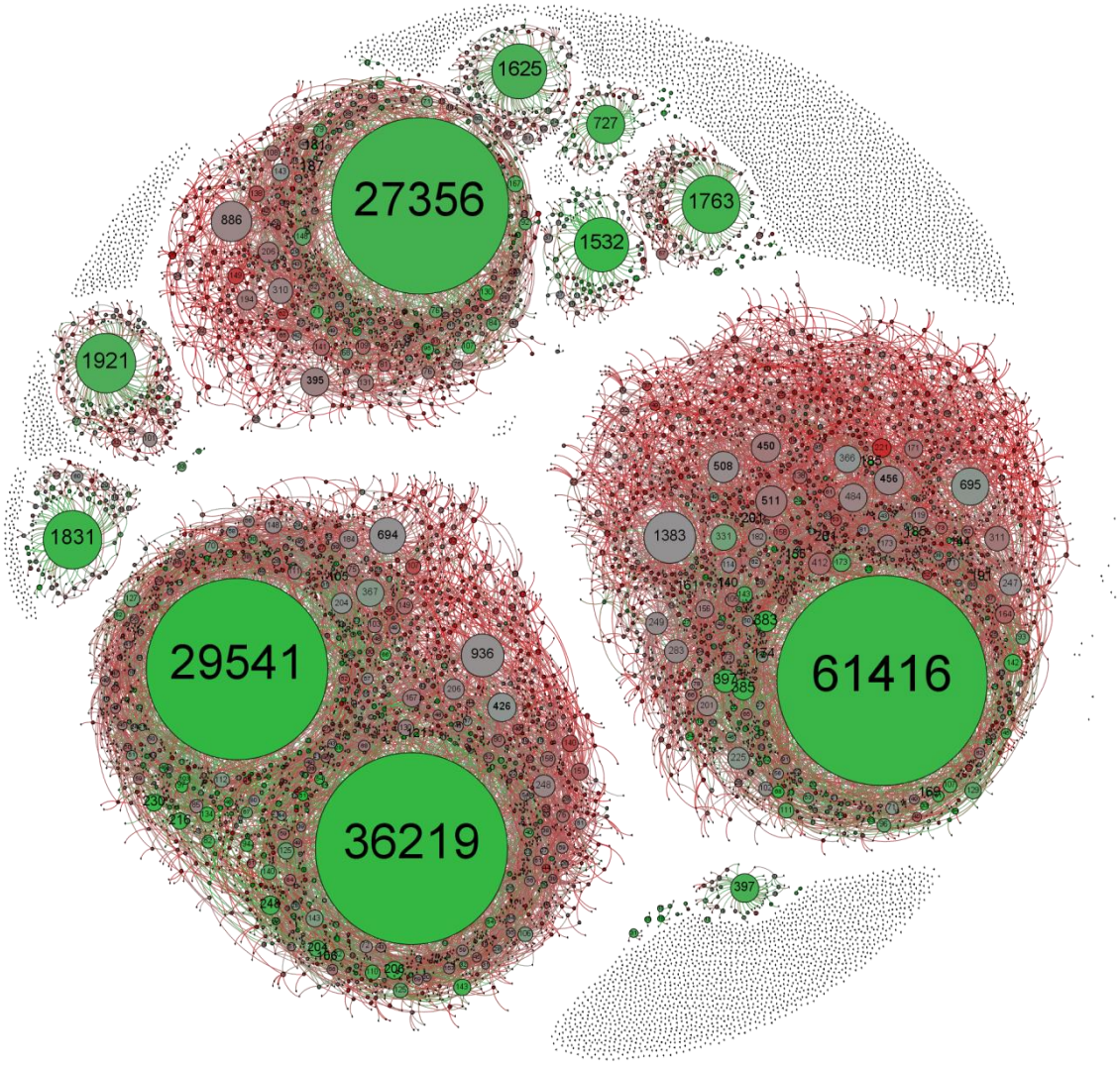



**Figure 6: A scarred mother was crossed with a homozygote (containing no scar sites). The resulting embryos, of which the scars can be seen as a single germ-cell of the parents was sequenced. This resulted in reads from which the Hamming graph shown here was constructed. Each node indicates a unique sequence. The size of the nodes indicates how many reads contain this sequence. The colors of the nodes indicate the average overall confidence value. The lower left two sequences are only a single Hamming distance apart from each other and share the same alignment to the reference. The graph was visualised in Gephi[9] using the Force Atlas 2 graph layout plugin[10].**

It is possible that two biological sequences are close in edit distance (Figure 6). Collapsing all reads member of a connected component into the most abundant sequence in the connected component will thus not yield proper results in all cases. In order to correctly collapse erroneous reads a method is required which is able to find the biological sequence a read most likely belongs to.

The base-calling confidence values between two sets of sequences can be used to find and assign reads to biological sequences. As the difference between the edge-connected reads is only a single base, the influence of GC percentage, homopolymers and other motifs which affect the confidence score will be very similar for both groups of reads. On average the base-calling confidence of the erroneously called base is lower than the correctly called base. In Figure 7 a distribution is shown

which enforces this claim. The distribution also shows that comparing the reads in individual cases will not separate the two groups.

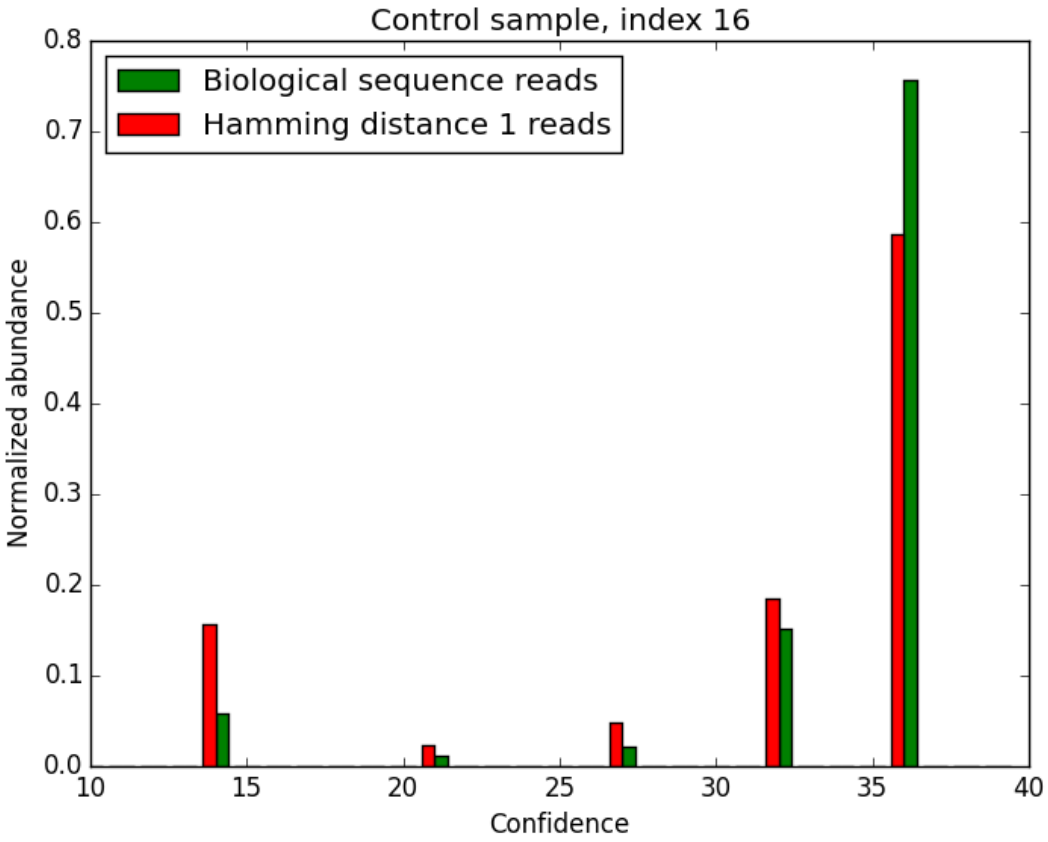



**Figure 7: The distribution of phred scores at position 16, for both correct reads and reads which have a different base called at position 16. Many of the available phred scores (1-40) are never assigned by the sequencer. Sample shown is a DNA amplicon sequencing dataset of a Cas-9 protein injected zebrafish embryo without injecting a sgRNA. Filtering individual reads based on their phred scores will shrink the abundance of the erroneous sequences but not remove them entirely.**

The confidence value averages can be incorporated into the Hamming graph by replacing the undirected edges with directional edges. The direction of an edge is decided upon by comparing the average confidence values responsible for the Hamming distance of 1 between the two sequences. When the absolute difference exceeds a threshold a directional edge is placed which points to the most confident sequence . When the absolute difference is smaller than the threshold, the edge is removed. Applying this methodology to all edges in the undirected Hamming graph reveals the biological sequences as nodes which are located centrally in connected components in the graph.

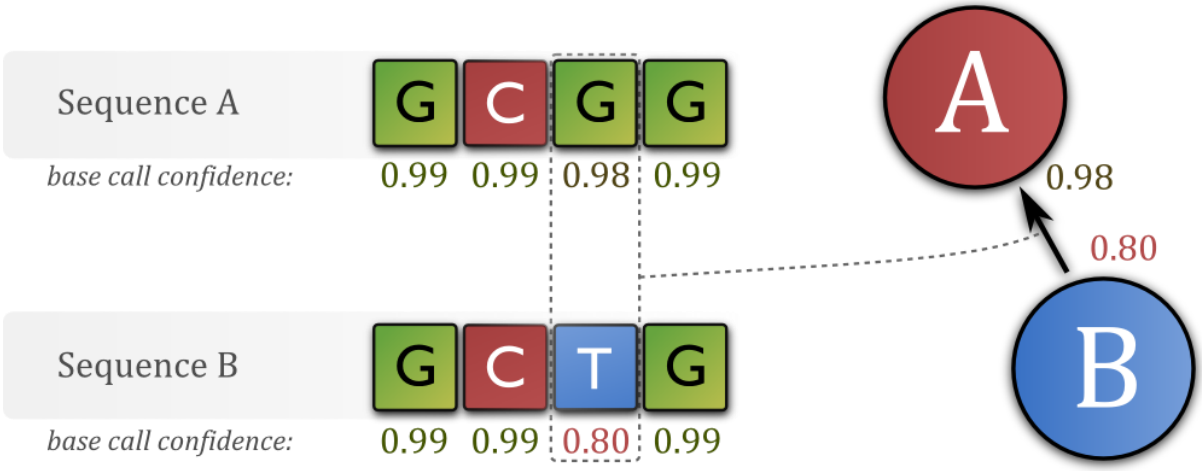



**Figure 8: Directed Hamming graph. Sequence A and B are separated by one Hamming distance, therefore an edge is positioned between A and B. The edge is directed to the on average most confident sequence for the base-call which causes the Hamming distance of 1.**

### Collapsing the directional Hamming graph

The directional Hamming graph is used to assign erroneous reads to real biological sequences. The collapsing process is performed iteratively, starting from the leaf nodes of the graph. The leaf nodes, which we define to have an in-degree of zero and out-degree bigger than zero are candidates for collapsing. All neighbors of the selected leaf-node which have a higher or equal abundance compared to the leaf node receive the reads assigned to the leaf node uniformly. After re-assignment, the amount of reads assigned to the leaf node will usually be zero and the leaf-node is removed from the graph. This process is executed iteratively until no leaf-nodes are left in the graph.

As error rates of about 10% are expected, collapses resulting in error rates higher than 100% (or another customizable expected error rate) are not collapsed. When no nodes with an in degree of 0 are available in the graph this can indicate two situations; either the collapsing is finished, or the directed hamming graph has a topology which prevents further collapsing. An unresolvable topological structure is a cycle which by circular reference does not allow for collapsing. To continue collapsing, the biggest cycle in the graph is detected using Johnsons algorithm[11]. Then the weakest edge, which indicates the smallest absolute difference in quality scores between nodes, is removed. This can change the topology allowing for collapsing and the algorithm is continued, resulting in a next collapse or removal of a weak edge. When no nodes can be collapsed and no cycles are left in the graph the algorithm stops and the sequence collapsing is considered finished.

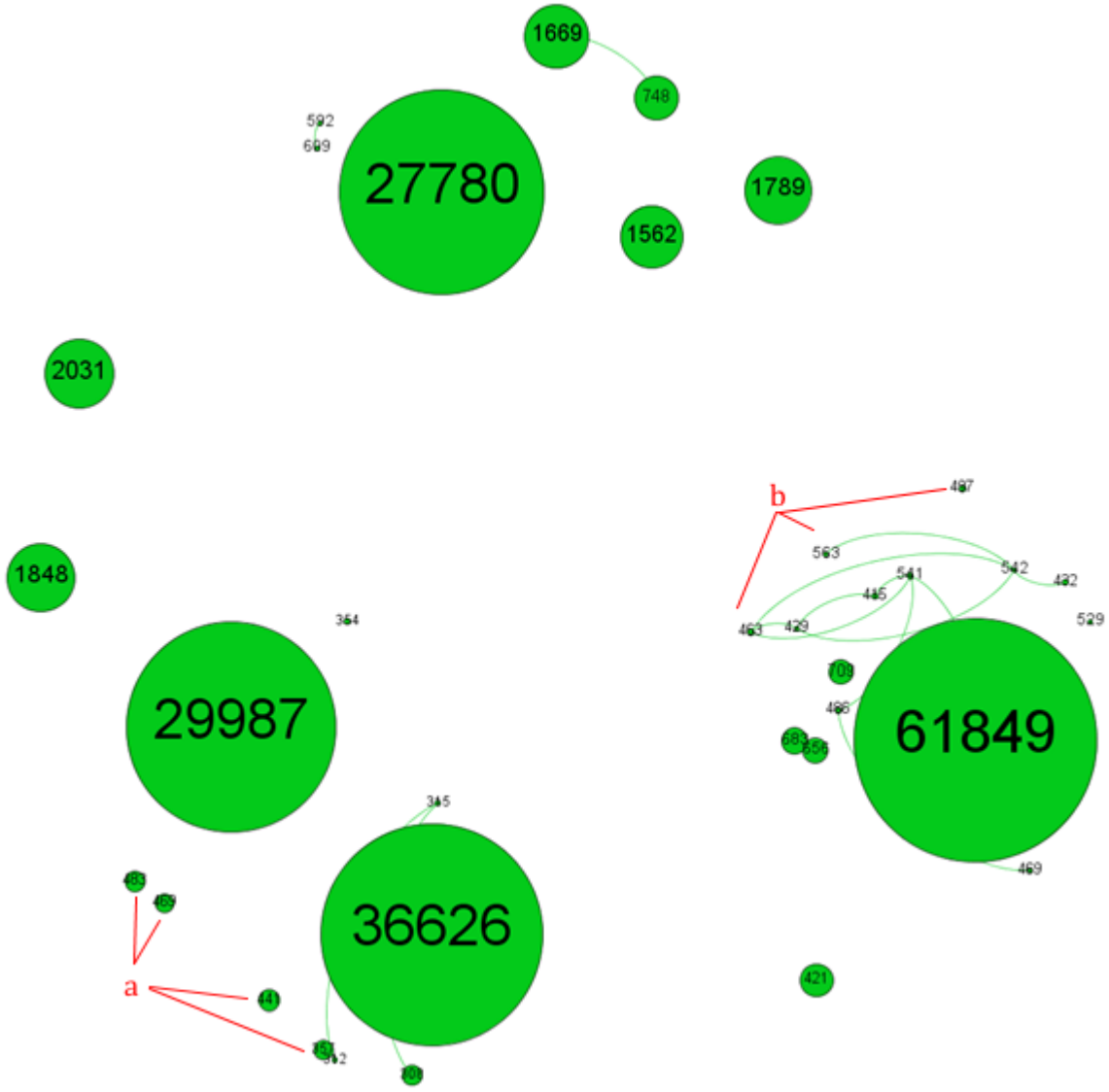



**Figure 9: Example of a collapsed dataset. The centers of components are kept. [a] Examples of sequences which are probably the result of PCR. [b] sequences which are not removed, these sequences attract many sequences, their error rate (reads collapsed/reads observed) is very high.**

The collapsed graph usually contains many lowly abundant sequences which have to be removed. Most of these sequences are not connected in the un-collapsed Hamming graph. A threshold is required to define the least amount of (collapsed) reads required for a marker to be called. The Hamming graph visualization can be used to make an informed decision on the value of this threshold such that real biological markers are kept while the singleton noise is removed (Figure 9).

Calculation of a complete Hamming distance matrix is computationally very expensive; every unique sequence needs to be compared to all other sequences, resulting in $(N^2/2 - 1)$ comparisons required to construct the distance matrix. High Hamming distances are discarded resulting in a sparse distance matrix, which used for graph construction. The use of a sparse matrix drops the high space-complexity of pairwise comparison, the high computational time complexity remains.

## Hamming graph approximation

Here we propose a method which does not require the pairwise comparisons between sequences while yielding a good approximation of the exact sparse hamming distance matrix, allowing for generation of Hamming distance graphs for bigger datasets. The lineage tracing datasets have specific characteristics which can be used to quickly approximate the sparse hamming distance matrix. The marker sequences are short and all the same size, secondly the sequences are often close in hamming distance to other sequences. Due to the nature of sequencing errors we know that most less-abundant nodes will be close in edit distance to big nodes. Edges considered as less important are edges which lie between rare (not abundant) nodes. Such edges will affect the collapsing result not at all, or at most marginally. The proposed method guarantees that there is no loss of important edges in the Hamming graph, while the Hamming graph approximation is fast.

A datastructure is filled which contains the abundant sequences present in a sample. Upon completion of this structure the Hamming distances can be extracted without performing any pairwise comparisons. Secondly, the distance of sequences not present in the data structure can be looked up in constant time. The limitations of this method include that between nodes in the datastructure distances of at most 2 substitutions can be found. The maximum distance which can be looked up for a sequence not present in the tree is a single substitution, this is enough for the purpose of Hamming graph construction. Due to the space-complexity of this method it is only practically applicable to short sized sequences.

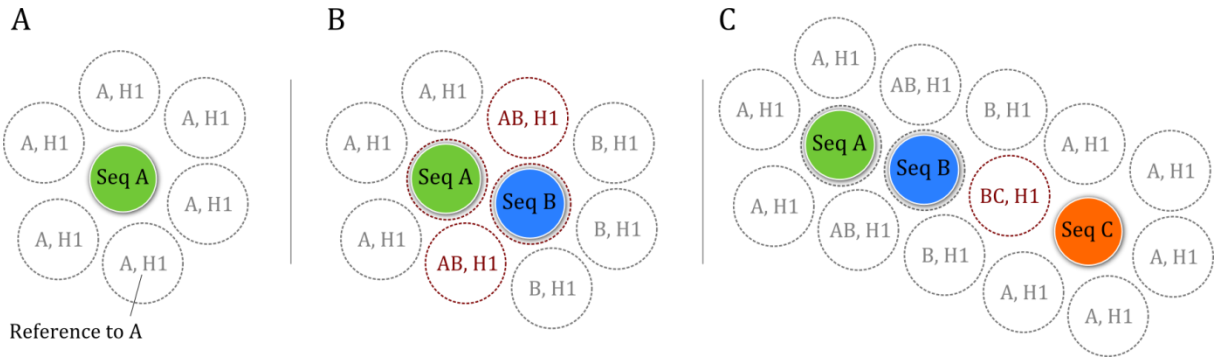



**Figure 10: Cartoon of Hamming distance approximation method used. (A) For every sequence added, all sequences 1 Hamming distance away to this sequence are added too. (B) When two sequences a single edit away are added, these will overlap in space and it is immediately known that the sequences are separated by single Hamming distance. To check whether sequence B is H1 away from any sequence which was already added, it suffices to only check if the space of B is already occupied, without adding B. (C) When a sequence two Hamming distances away (C) is added, the sequences will overlap, but only by their H1 away sequences.**

For fast lookup and comparison of sequences, all sequences are stored in a tree. As there are four bases available for every position in a sequence, a straightforward way to store the sequences is a quadtree. The tree has a depth matching the sequence length. Every node in the quadtree depicts a base, the depth of the node matches the position of this base in the sequence. Every leaf node stores references to the reads which contain the sequence which reach it. The quadtree allows for a fast, constant time look-up and insertion of sequences.

The quadtree allows for extraction of Hamming distances of 1 or 2 without doing pairwise comparisons by storing all H1 distant sequences of every sequence which is added to the tree. For the sequences at H1 from the sequence a reference to the sequence added is stored. When a pair of sequences is H1 apart, two leaf nodes will contain references to both sequences as shown in Figure

10 and Figure 11. When a leaf node contains a reference to an edited sequence and a verbatim sequence this means the sequences are H1 apart. When a tree contains a leaf node with references to two permuted sequences and no leaves with these sequences together with one being a permuted, the sequences are H2 apart.

The quad tree does not need to be fully populated by all sequences. Sequences with low read counts can be looked up in the quadtree without adding them (Figure 10b). This results in finding the hamming distances to sequences which are present in the tree and misses distances to sequences which are not present in the tree. This greatly increases the speed of the algorithm without losing distances which are important.

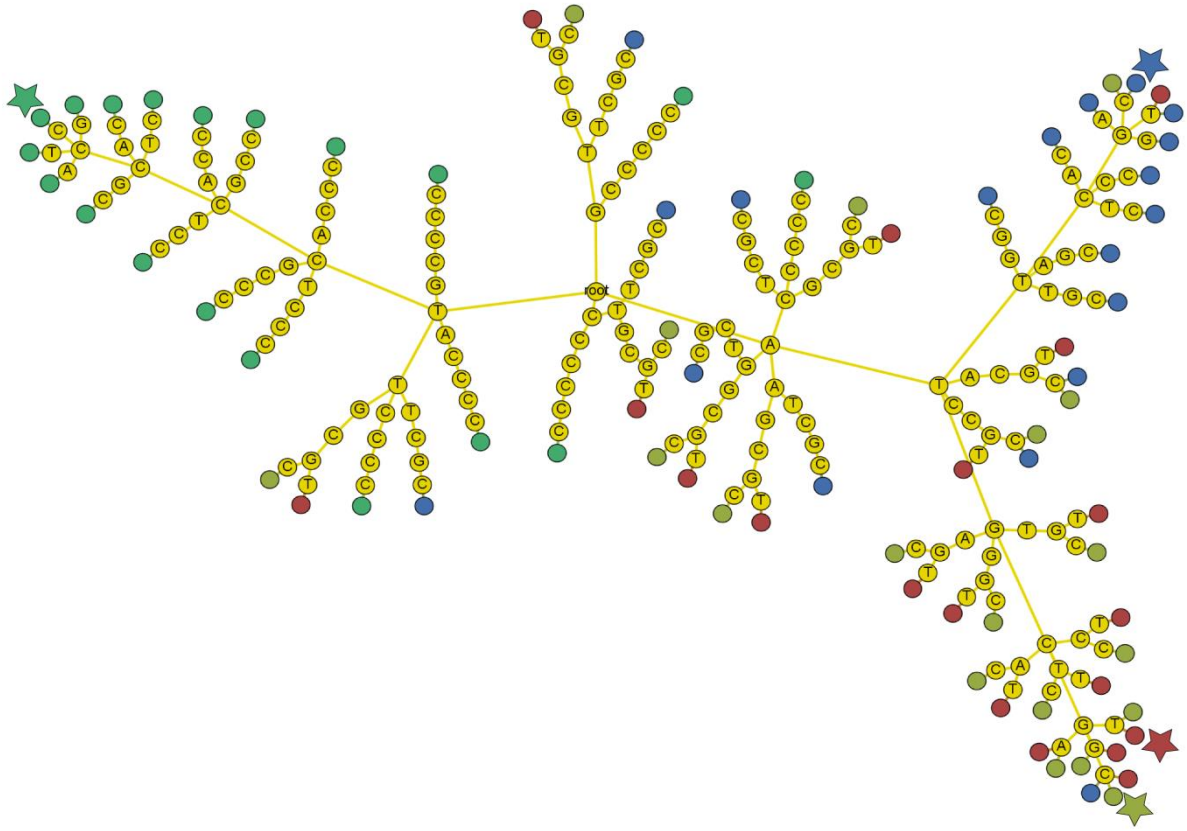



**Figure 11: Quadtree datastructure used. Example of a generated quadtree for four short sequences. Stars indicate verbatim sequences, unmarked leaf nodes indicate H1 distant sequences. Sequences in tree: ATGCGC, ATGCGT, ACGCGC, TCCCCC. Note that in reality the non-yellow nodes are not part of the edit graph, they are just there for visual reference of the edit state and sequence. The ancestors of the colored nodes are the leaf nodes of the quadtree.**

## Comparison

Validation of the method is required to find problems and to demonstrate its capabilities. Multiple ways of comparing the method are available. First it is possible to run the method on a dataset where the real biological sequences are known. Such datasets are hard to obtain, one example used is the control dataset where no sites where marked, resulting in a dataset containing a single biological sequence. In that example the algorithm performs worse than when alignment as descriptor. In order to obtain a better performance than using alignment as descriptor, the indel errors should be corrected and no false positives should be generated. In the proposed algorithm the indel errors are not corrected, and all sequences share the same alignment. For the data shown in Figure 4, three sequences in the *Wt* component are called erroneously.

The original scartrace method uses alignment of the reads to the reference construct sequence[3]. The resulting alignment can be expressed as a CIGAR string, which is used as identifier for the marker. For every sequence with a membership higher than 5% for a CIGAR string a new marker is called. The original-method assignments are compared with the ones of the collapse in Figure 12

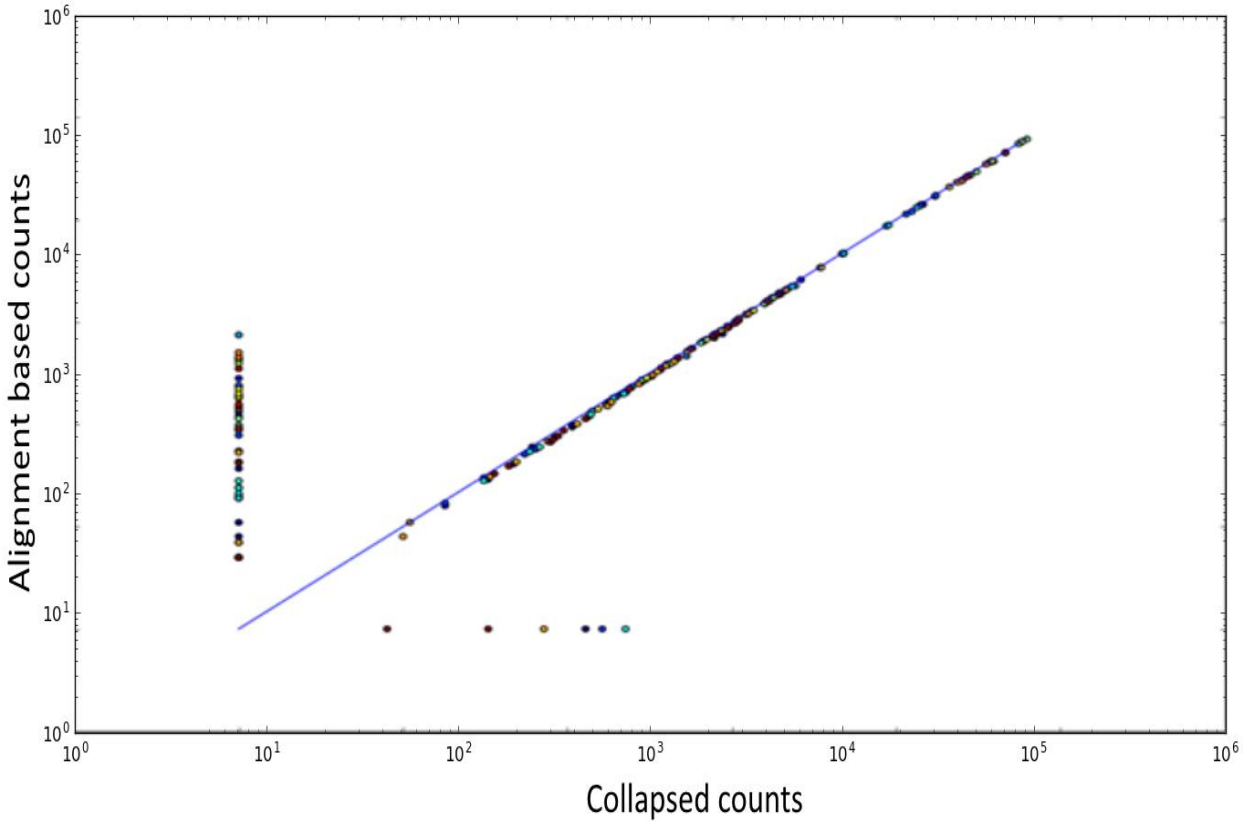



**Figure 12: Counts resulting from collapsing compared with counts resulting from alignment. The blue line indicates an identical call. For the high frequencies the methods produce very similar results, the collapser almost always assigns a higher count to these scars. For lower read counts the collapser removes more counts than the alignment method does.**

## Validation by read-simulation

To perform in-silico validation, amplicons where simulated by using the amplicon simulation feature in the read-simulation software package ART[12]. The sequences used as simulation target which correspond to correct biological sequences where chosen to be 1 Hamming distance apart, in order to construct a complex situation. Error models used for amplicon simulation where chosen to match the sequencer used to sequence the scars in-vivo. Analysis of the resulting Hamming graph reveals that this graph is very dissimilar from in-vivo experiments (Figure 5). There are less erroneous reads and the erroneous reads show no clear difference in abundance as opposed to the in vivo reads. Lastly the phred scores simulated by the read simulator seem to be too accurate. All these features lead to severe overestimation of the algorithm's performance when applied to such an artificial dataset.

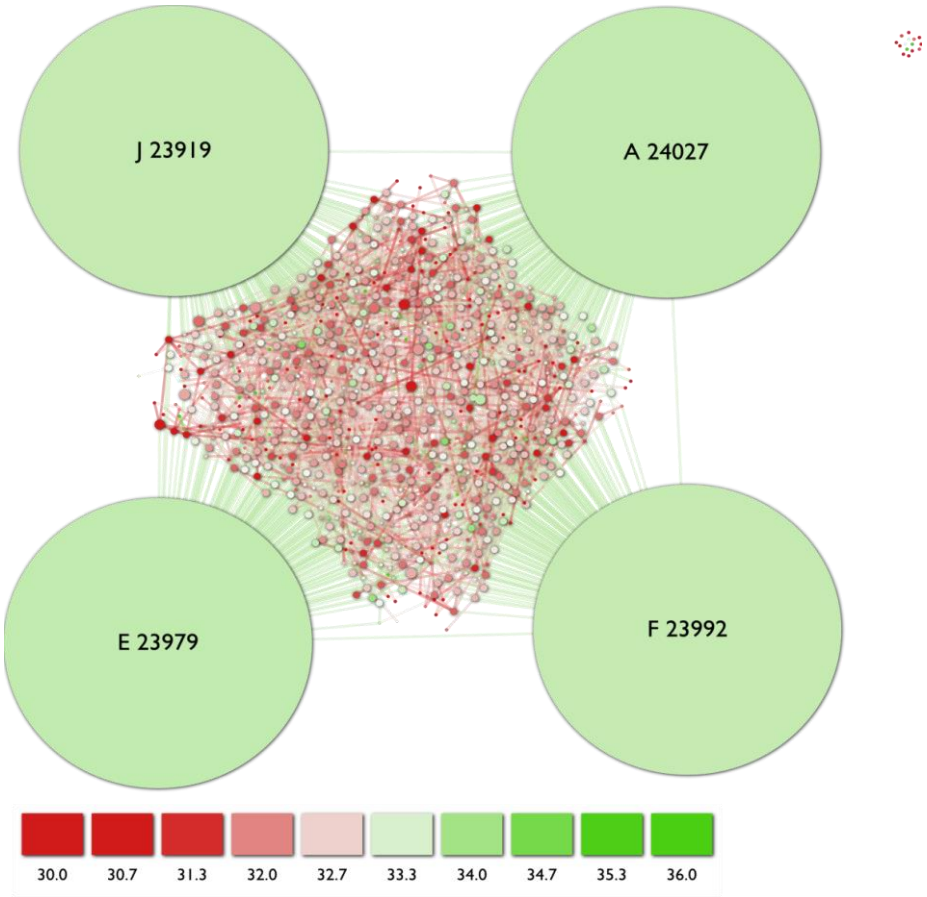



**Figure 6: Hamming graph visualization of reads simulated using the ART amplicon simulation tool. Reads for four sequences are simulated, these sequences are single Hamming distance away to two others. The scaling of the nodes is exactly the same as the other plots (absolute surface scaling). Almost no unconnected sequences are present and no relatively big erroneous nodes. The range of phred scores is much smaller than observed for the in vivo samples. Collapsing yields only the 4 initial nodes after removal of all sequences with abundance < 3 reads.**

### Collapser discussion

In the collapsing approach insertion and deletion errors are not taken into account, the effect can be quite significant for very abundant sequences. Incorporation of indel errors is challenging because there are no confidence values which relate to these errors. The implication of this is that the least abundant sequence of a set of sequences, with a relative difference bigger than the indel error-rate (approximately 0.1%) is impossible to detect reliably. A possible solution would be the calculation of edit distances between the graph components. The edit distances can then be used to check whether the observed frequency of one of the two components can be explained by the frequency of the other multiplied by the expected indel error-rate.

The Hamming graph approximation algorithm becomes memory hungry for big fastq files (fastq files of multiple gigabytes). The memory footprint can be reduced by splitting the quadtree into multiple parts by configuring the algorithm to check a single quadrant, or even sub-selections of a quadrant at a single time. This cannot be done indefinitely to obtain infinite sub-problems without any cost in time; for solving every one of these sub-problems the full list of sequences has to be traversed. Using this method it is possible to perform collapsing of all reads of an experiment, instead of performing collapsing per sample.

## Simulation of scarred cell-lineage trees

Simulated cell-lineage trees can help in understanding cell-lineage tracing and allows for benchmarking of algorithms. In the simulation a cell is simulated as a vector of length S, where S indicates the maximum amount of lineage markers. In case of using mutations as lineage marker, S is the amount of bases in the genome. In case of scartrace, S is equal to the amount of targeted integrations, in the simulations we set this value to be 4. Cells divide with a fixed probability $p_d$ until the desired amount of leaves is reached. When $p_d$ is set to 1 and the cells are a power of 2, the resulting lineage tree will be balanced. In reality cells which have died are also leaves of the cell lineage tree, but this edge-case will be omitted in this simulation as these cells are not sampled in-vivo.

In the simulation the decision whether a scar is introduced is decided upon by a fixed probability. This aims to capture the event of a double strand break being induced by Cas-9 and erroneous repair resulting in a scar. The scarring probability is estimated by experiments where a complete embryo was sequenced after a known point in time. For the simulations shown here the probability used is 10%.

When a scar is selected to have occurred, a scar value is decided upon. Many dynamics-experiments have been conducted. In these experiments a complete embryo is sequenced at a known point in time. Based on these experiments the probability of each scar ($\alpha_i$) is established. In this case $\alpha_i$ is coarsely estimated by taking the average ratio of scar i over all scars observed over all the dynamics experiments.

In the simulation the scar probabilities are modeled as a discrete random variable which allows for sampling. In this model the scars are sampled with replacement. When sampling from $\alpha_i$ without replacement, all scars in the lineage tree will be unique, simplifying the resulting inference problem. The correct cell-lineage tree *T* can be extracted from the simulated lineage tree by collapsing all cells

in which no new scar was introduced into their parent (exact algorithm in supplement). The maximum traceable depth of the lineage tree which can be achieved is equal to *S*.

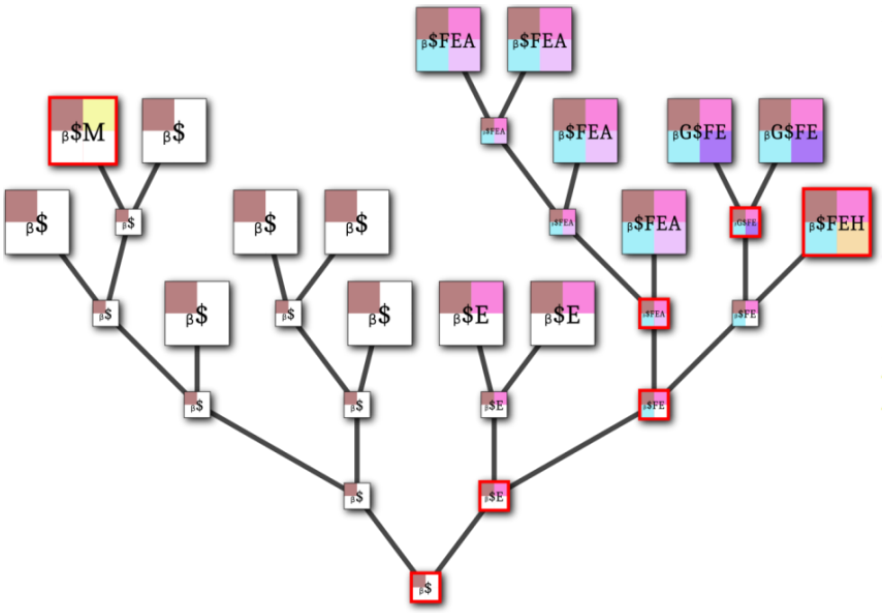



**Figure 13: Simulated cell-lineage tree. Boxes indicate cells, the leaf nodes indicate cells currently present in the organism. Each of the cells has 4 scar sites. A white scar-site state indicates that the site is unscarred *(Wt)*. $ is a scar which occurred in the single cell stage, and is inherited by all cells. The scars in this three are sampled without replacement, resulting in no duplicate scars. Red stroked cells indicate that a new scar is introduced.**

Artificial samples are generated by random sampling from the leaves of the simulated lineage tree. The lineage marker counts from the sampled leaves are summed resulting in a frequency matrix. The scar-site information is not recorded reflecting the in-vivo experiments. Result from the simulation is a frequency matrix *F*, and the correct lineage tree *T* (Figure 16). In reality most types of scartrace samples are spatially separated, for example slices or complete organs. The amount of lineages present in samples in-vivo be lower than when random sampling from all leaves.

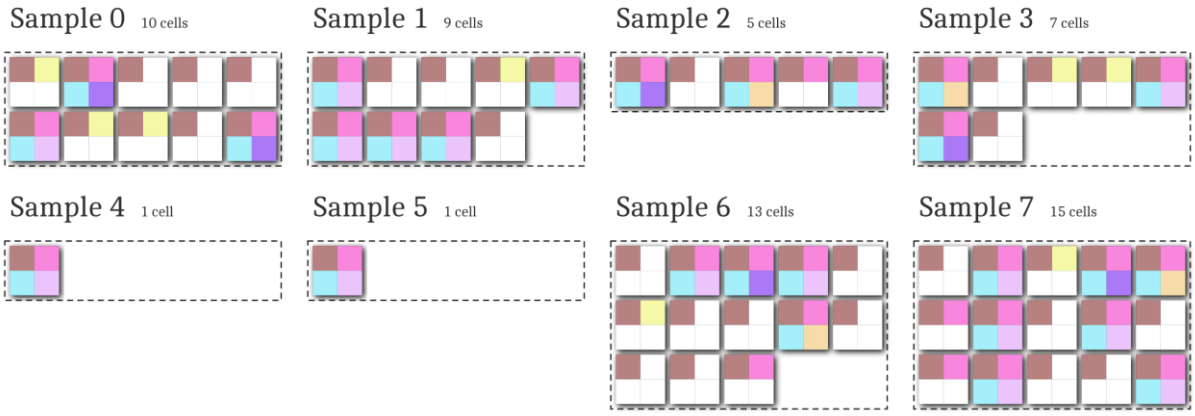



**Figure 14: Artificial samples from the simulated lineage tree. Each box indicates a cell, as in Figure 14. The samples are generated by sampling from the leaves of the artificial lineage tree by random uniform sampling.**

| | A | E | $ | G | F | H | M |
|---|---|---|---|---|---|---|---|
| Sample 0 | 0.05 | 0.14 | 0.45 | 0.09 | 0.14 | | 0.14 |
| Sample 1 | 0.20 | 0.20 | 0.36 | | 0.20 | | 0.04 |
| Sample 2 | 0.07 | 0.27 | 0.33 | 0.07 | 0.20 | 0.07 | |
| Sample 3 | 0.06 | 0.17 | 0.39 | 0.06 | 0.17 | 0.06 | 0.11 |
| Sample 4 | 0.25 | 0.25 | 0.25 | | 0.25 | | |
| Sample 5 | 0.25 | 0.25 | 0.25 | | 0.25 | | |
| Sample 6 | 0.07 | 0.19 | 0.48 | 0.04 | 0.15 | 0.04 | 0.04 |
| Sample 7 | 0.13 | 0.23 | 0.38 | 0.03 | 0.18 | 0.03 | 0.03 |

**Figure 15: Frequency matrix based on the artificial samples, frequencies are calculated for each sample by dividing the sum of observed scars by the total amount of observed sites (total number of reads for that sample).**

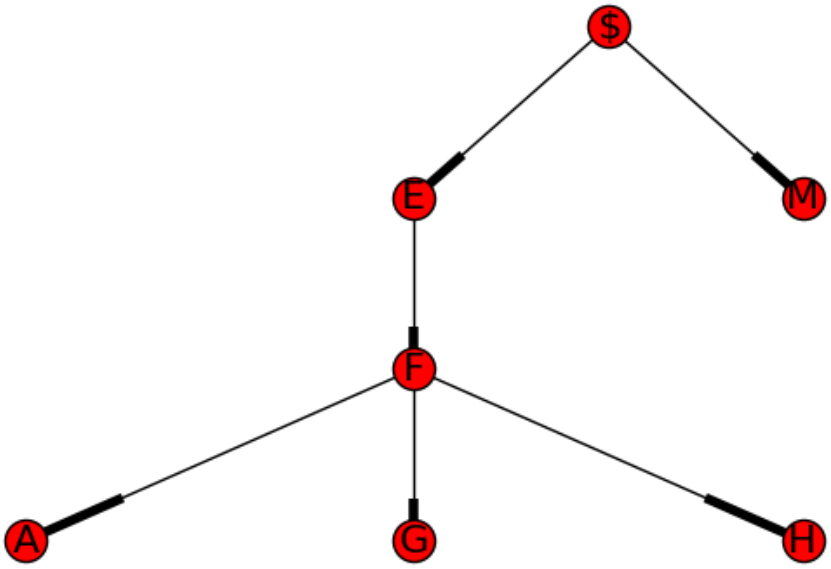



**Figure 16: Correct lineage tree, given the simulated lineage tree shown in Figure 13. Each node indicates a newly introduced scar. A cell has the cumulative sum of the scars present in the tree. For example, there only exists a cell with scar H with F and E also present in its scar-sites. Scar $ is the driver mutation, which can be artificially added if not present in the dataset.**

## Reconstruction of a lineage tree given experimental data

To reconstruct a tree given single cell measurements, there are multiple possible methods. The simplest is to find a tree which explains the observed data with the least amount of edits. Selection of a tree using this criterion is called maximum parsimony optimalisation[13–16]. Obtaining a tree for such a case can for example be achieved by using Neighbor Joining[17,18]. In most of the currently performed experiments the measurements are not single cell measurements. Instead, the lineage-marker frequencies are derived from a pooled mixture of many cells. The scar site and originating cell are not measured directly when using mixtures of cells, and thus the maximum parsimony criterion cannot be applied.

There are computational methods available which are able to deal with mixed lineage samples based on mutations[19]. Such methods mostly originate from cancer-biology where the cell-lineages and their corresponding driver mutations help in understanding cancer progression. In cancer lineage tracing, the cell-lineages are identified using mutations in the genome, which are often assumed to be non-over writable by the infinite sites assumption. In the scar-trace method the sites are also assumed to be non-over writable[3]. Instead of using mutations as lineage marker, scars are used. The main difference between the data resulting from the two methods is that in scar-trace it is possible that the same scar occurs multiple times, with a predictable rate. When only scars are used which are introduced once, normal mutation based lineage inference algorithms apply. Here we will build upon the *error free* lineage tracing algorithm by Mohammed El-kebir[19].

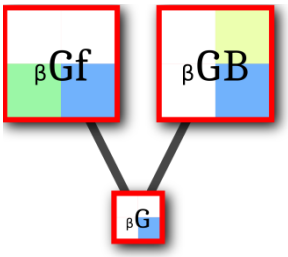



**Figure 17: Small example lineage tree of two cells. The founding cell is the lineage tree root, and has divided into two cells. The checkerboard pattern in the cells shows the state of the 4 scar sites in the cells. Scar G was introduced first, B and f later. When sampling from the leaves of this lineage tree, G is always observed in higher or equal frequency than B. This means G was introduced before B.**

### The sum rule

The basis of most lineage tracing methods is that the sum of frequencies of a descending lineage marker can never exceed the frequency of an ancestral marker. For example the lineage tree in Figure 17 shows scar G was introduced before the cell gave rise to the two leaf cells. G will be measurable in both cells. B and f are introduced later. When sampling from the leaves of this tree, the measured frequency of G will always be higher than f and B. To be more precise; f and B summed will equal G. This property is called the sum-rule, and can be used to reduce the size of the lineage inference problem drastically, but usually does not yield a definite result[19].

To obtain the lineage-marker relations a directed graph structure, the ancestry graph *A* is used (Figure 18). The edges in *A* contain all ancestral relationships in the dataset. In this graph every node reflects a unique lineage marker, in this case a scar. By definition every scar is only present once in A. The edges represent ancestral relationship between scars. The graph is constructed by checking for every node if the scar is more abundant than another in all samples (rows in *F*). If this is true an edge

is positioned between the two scars. The true lineage graph *T* is an arborescence of $A^{19}$. In practice this means that picking an arborescence means choosing a root node, enforcing that nodes will have one incoming edge and all nodes are visited. Not all possible arborescences are candidates for *T*. Some arborescences will violate the sum-rule, meaning that the descendants of one or more nodes have a sum which combined, is higher than the sum of their parent. A perfect *A* has edges between a scar and all its predecessors (Figure 18a). Mixed samples induce edges in A where the edge represents ancestry which does not exist (Figure 18b). The spurious edges in the observed *A* also produce more arborescences to evaluate.

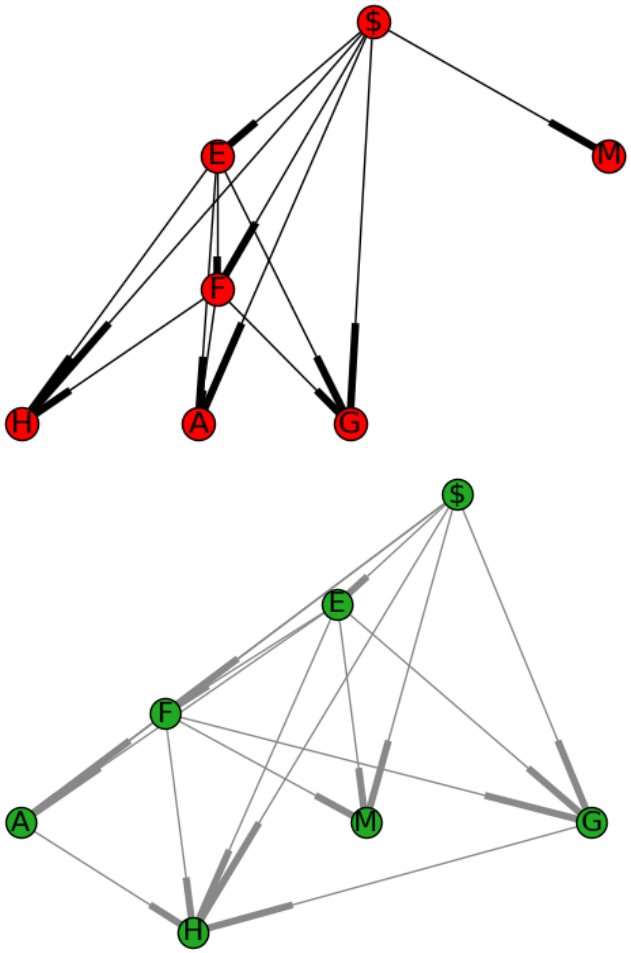



**Figure 18: Ancestry graphs, these show the frequency relation between the observed lineage markers. High abundant lineage markers indicate an appearance early in time. A lineage marker is connected to all its descendants. Top: expected ancestry graph of the simulated lineage tree shown in Figure 13. The expected ancestry graph is generated by placing edges from every scar to all its successors. Bottom: observed ancestry graph based on simulated mixed samples. This graph contains all edges in the expected ancestry graph, due to mixing it also contains edges which do not reflect real ancestral relationships.**

### Artificial lineage root

In the original E-K method a driver mutation is required to function as root node of the lineage tree. This requirement is reasonable as cancer usually does start as result of a driving mutation[20]. To be able to reconstruct a lineage tree based on scar data, a unique mutation is required in the founding cell. This mutation is conveniently enforced in the simulation example, as scar "$". In vivo this driver mutation cannot be enforced, and is added artificially. The artificial frequency $ should reflect a frequency where every cell is assigned the scar once. In the lineage-tracing experiment every cell produces $Wt_{Reads}$ and $Scar_{Reads}$. To obtain a frequency which matches the frequency of a single scar site, the total amount of reads of a single sample is divided by the amount of scar-sites *S*. The frequency of the artificial scar is thus set to be ($Wt_{Reads}$ + $Scar_{Reads}$) / *S*.

### Iteration over all arborescences

For every other node *i* in *A,* an incoming edge needs to be selected. Let $i_{out}^{k}$ be the amount of possible incoming edges for every node *i* in the total set of nodes N. The total amount of arborescences is the product of the amount of possible incoming edges for all i: $\prod_{k=0}^{N} i_{out}^{k}$.

Consider an ancestry graph, in matrix notation:

| | A | C | B | D | I | H | K | J | O | N |
|---|---|---|---|---|---|---|---|---|---|---|
| A | y | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| C | . | y | . | . | x | . | x | . | . | . |
| B | . | . | y | x | x | x | x | . | . | . |
| D | . | . | . | y | . | . | x | . | . | . |
| I | . | . | . | . | y | . | . | . | . | . |
| H | . | . | . | . | . | y | x | . | . | . |
| K | . | . | . | . | . | . | y | . | . | . |
| J | . | . | . | . | x | . | x | y | . | . |
| O | . | . | . | . | . | x | x | . | y | . |
| N | . | . | . | . | x | . | x | x | . | y |

Y indicates the edge is present in the arborescence, an x indicates that an edge can be selected to be present. To select an arborescence every but one column needs to have a single Y (omitting the diagonal). All arborescences are iterated over by taking each of these combinations.

### Selecting sum-rule compliant arborescences

Arborescences are checked for compliance with the sum-rule by evaluating the sum-rule for every node. If a violation is found, the arborescence is discarded. The correct lineage tree *T* is present in the set of compliant arborescences[19]. In the E-K paper, every sum-rule compliant arborescence is considered a proper solution to the problem. By complete enumeration we find all of these arborescences.

Intuitively it makes sense to take all compliant arborescences and show the observed edge frequencies on *A (*Figure 19*).* While this does show the main lineage relations, the frequent edges can still be incorrect. The amount of times the edge is present in a compliant arborescence is shown, edges with high frequencies can be incorrect, while some edges with low frequencies, but higher than zero are present in *T*.

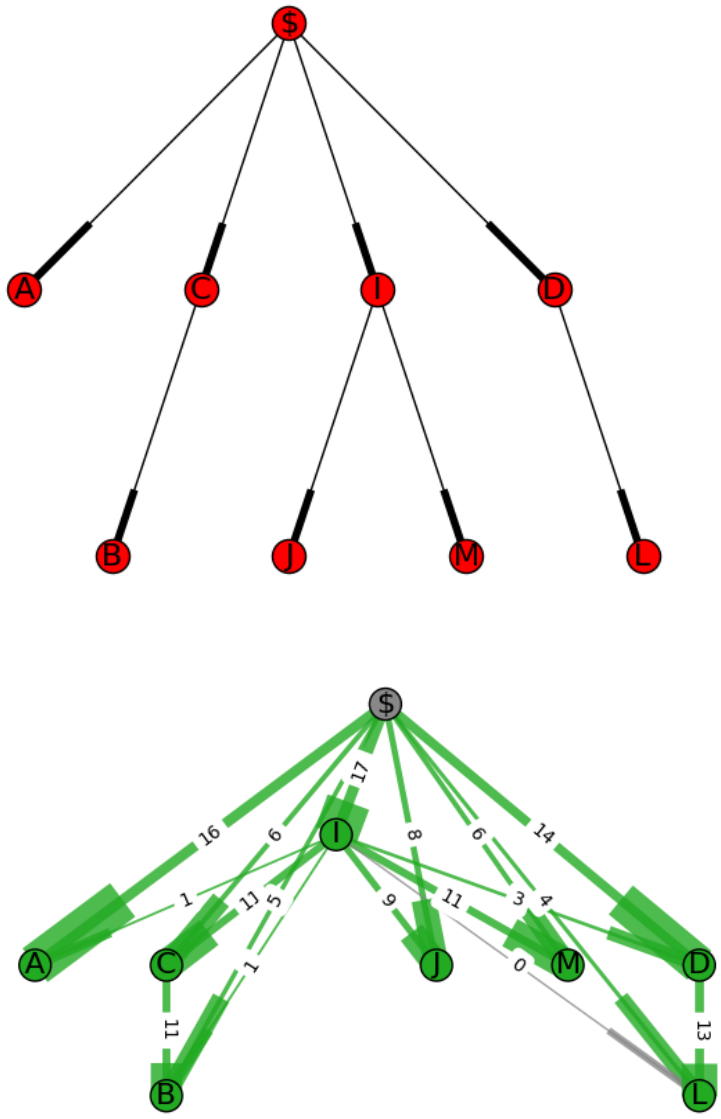
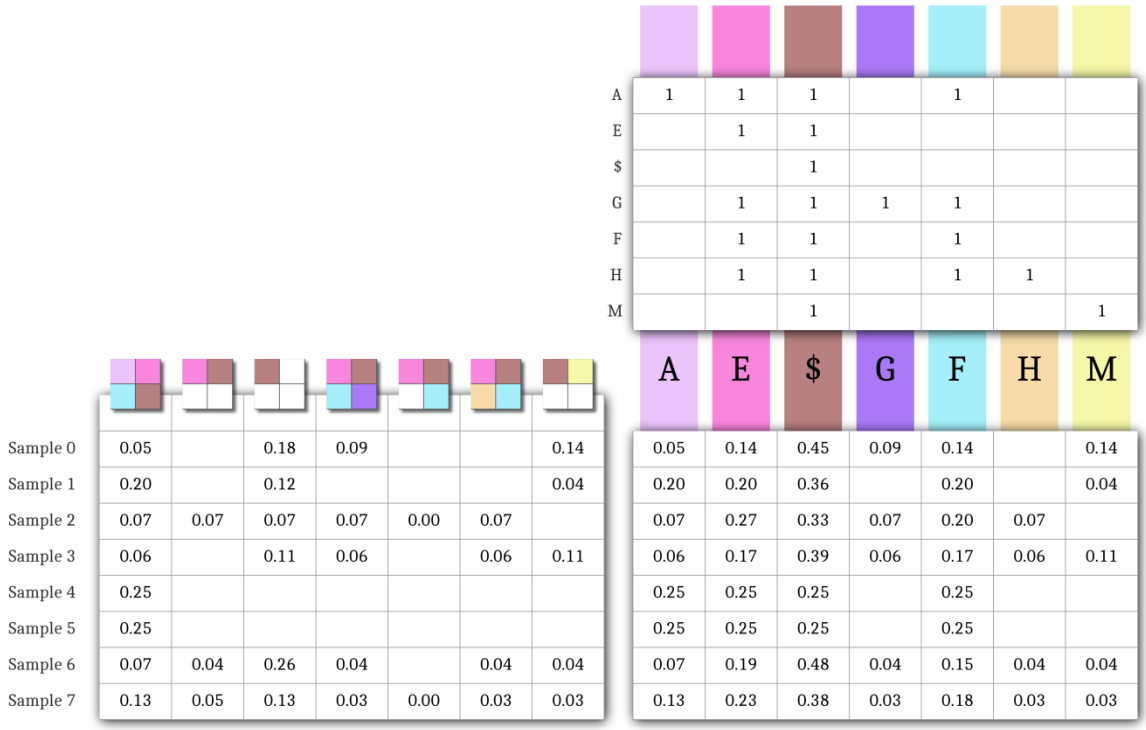



**Figure 19: Top: Simulated lineage. Bottom: Compliant arborescence frequency summation for the simulated lineage graph, the weight of every edge indicates the amount of sum-rule compliant arborescences which contain the edge. The relation I>C is observed more than the relation $>C while invalid. The weight 6 of edge $>C indicates that some of the sum-rule compliant arborescences indeed contained the correct relation.**

By selecting an arborescence, which is described in the clonal matrix B, the usage matrix, which contains the inferred cell frequencies can be calculated. In E-K it is shown that this matrix can be calculated by taking the *deficit* of each node in the ancestry graph. This is the amount of frequency which is not explained by the sum of the frequencies of the children of a node[19].

A small modification has to be put in place. In E-K mutations can never have a frequency higher than 0.5 due to the infinite-sites assumption combined with a human having two chromosomes. Therefore in E-K it is defined that the observed frequencies $F = \frac{1}{2}UB$. In scartrace the maximum frequency which can reached for a single uniquely introduced scar is 1. The definition of F can thus be simplified to $F = UB$. In Figure 20 the usage matrix, frequency matrix and the correct arborescence obtained from the simulated samples in Figure 14 are shown.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | | 1 | | |
| E | | 1 | 1 | | | | |
| $ | | | 1 | | | | |
| G | | 1 | 1 | 1 | 1 | | |
| F | | 1 | 1 | | 1 | | |
| H | | 1 | 1 | | 1 | 1 | |
| M | | | 1 | | | | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Sample 0 | 0.05 | | 0.18 | 0.09 | | | 0.14 |
| Sample 1 | 0.20 | | 0.12 | | | | 0.04 |
| Sample 2 | 0.07 | 0.07 | 0.07 | 0.07 | 0.00 | 0.07 | |
| Sample 3 | 0.06 | | 0.11 | 0.06 | | 0.06 | 0.11 |
| Sample 4 | 0.25 | | | | | | |
| Sample 5 | 0.25 | | | | | | |
| Sample 6 | 0.07 | 0.04 | 0.26 | 0.04 | | 0.04 | 0.04 |
| Sample 7 | 0.13 | 0.05 | 0.13 | 0.03 | 0.00 | 0.03 | 0.03 |

| A | E | $ | G | F | H | M |
|---|---|---|---|---|---|---|
| 0.05 | 0.14 | 0.45 | 0.09 | 0.14 | | 0.14 |
| 0.20 | 0.20 | 0.36 | | 0.20 | | 0.04 |
| 0.07 | 0.27 | 0.33 | 0.07 | 0.20 | 0.07 | |
| 0.06 | 0.17 | 0.39 | 0.06 | 0.17 | 0.06 | 0.11 |
| 0.25 | 0.25 | 0.25 | | 0.25 | | |
| 0.25 | 0.25 | 0.25 | | 0.25 | | |
| 0.07 | 0.19 | 0.48 | 0.04 | 0.15 | 0.04 | 0.04 |
| 0.13 | 0.23 | 0.38 | 0.03 | 0.18 | 0.03 | 0.03 |

**Figure 20: (left) Usage matrix, rows show inferred cell-compositions, (right-top) Clonal matrix of the selected arborescence, (right-bottom) observed scar-frequencies.**

## A real data-based proof of concept

In this experiment, described in the scartrace paper[3] a single cell embryo is injected with a synthetic guide-RNA and RNA coding for Cas-9 in the yolk. The embryo is grown into adulthood, and then slices of the fin are dissected and sequenced separately. The resulting reads are collapsed per sample, the collapsed frequencies are used further down the analysis. When naively generating the ancestry graph, the scars are ordered mostly based on their $\alpha_i$ value, as many of the scars will be introduced multiple times (Figure 21, Figure 22). To obtain an ancestry graph which reflects ancestral relations, the data is filtered to remove scars which might have occurred multiple times. Removal is performed such that the counts for all of the abundant scars are set to Wt. The filtering was performed such that $\alpha_i < 0.0001$. From the resulting scars we observed that the cells in the rows A,B and C are very similar, indicating that these cells probably originated from the same cell lineages (Figure 24).

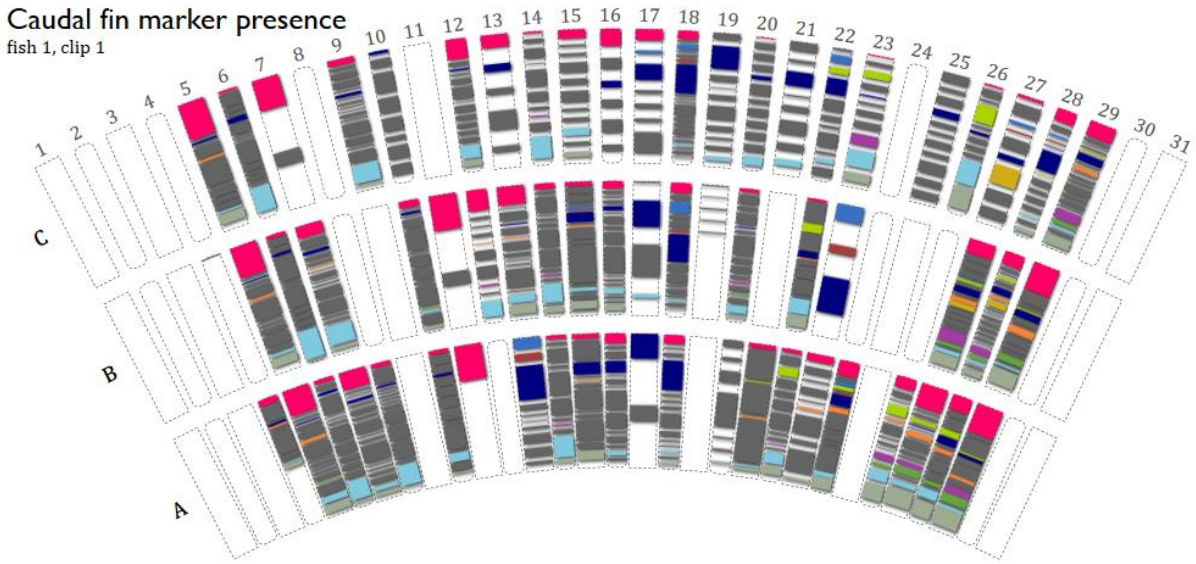



**Figure 21: Original scar distribution, collapsed marker frequencies are shown. Every dashed section indicates a sample, square samples are inter-ray pieces. Rounded samples are samples from the rays (bones).**

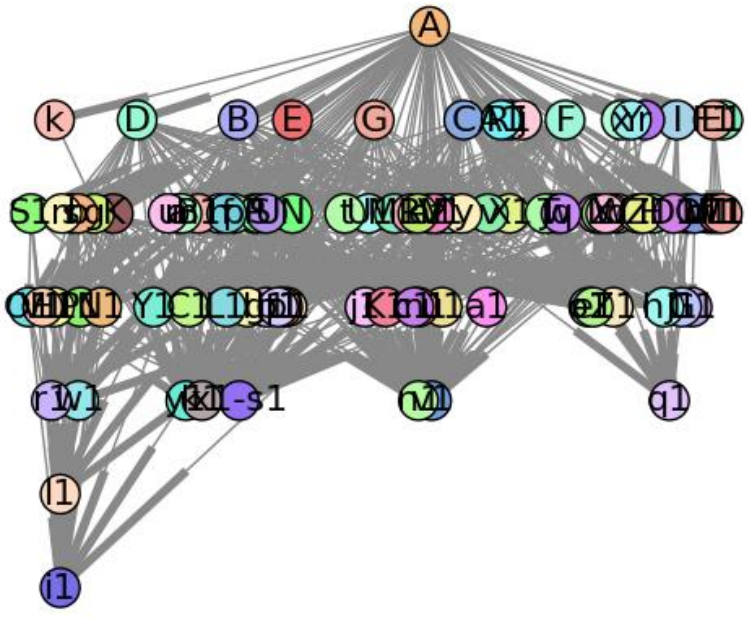



**Figure 22: Naively created ancestry graph based on the unfiltered collapsed scar frequencies (Figure 21). The hierarchy of the scars in this graph does not reflect ancestral relations. The ordering mostly reflects the probability that the scar is introduced ($\alpha_i$).**

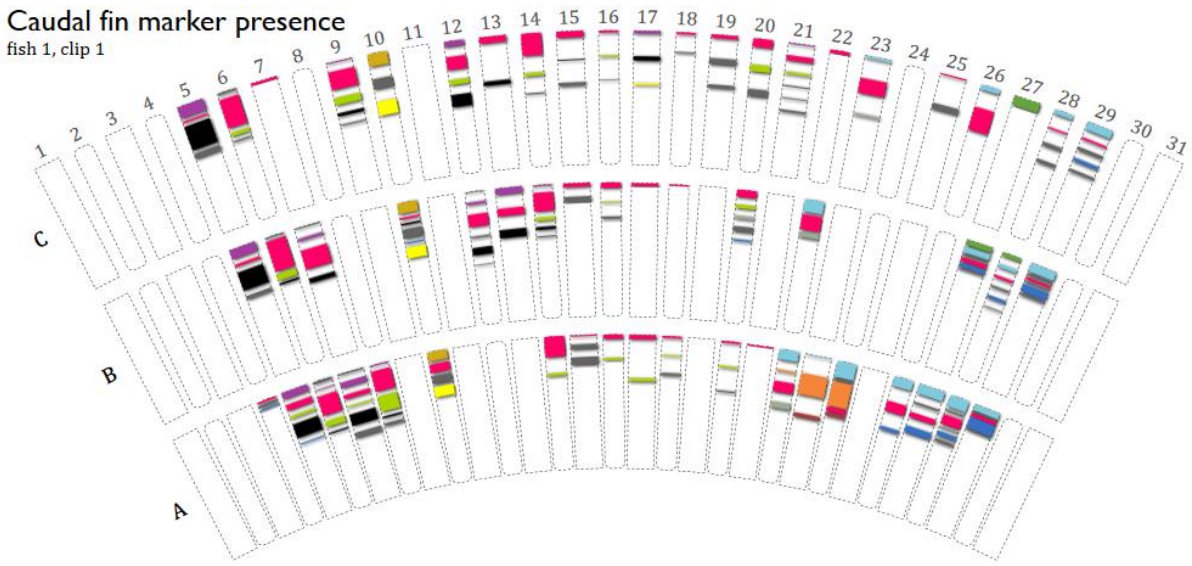



**Figure 24: Caudal fin lineage markers which are likely to be only introduced once based on their $\alpha_i$ value. As concluded in the original scartrace paper, there is a very strong correlation of scars in the same ray or inter-ray visible. Identically colored boxes indicate the same scar present, the size of the boxes indicate the abundance.**

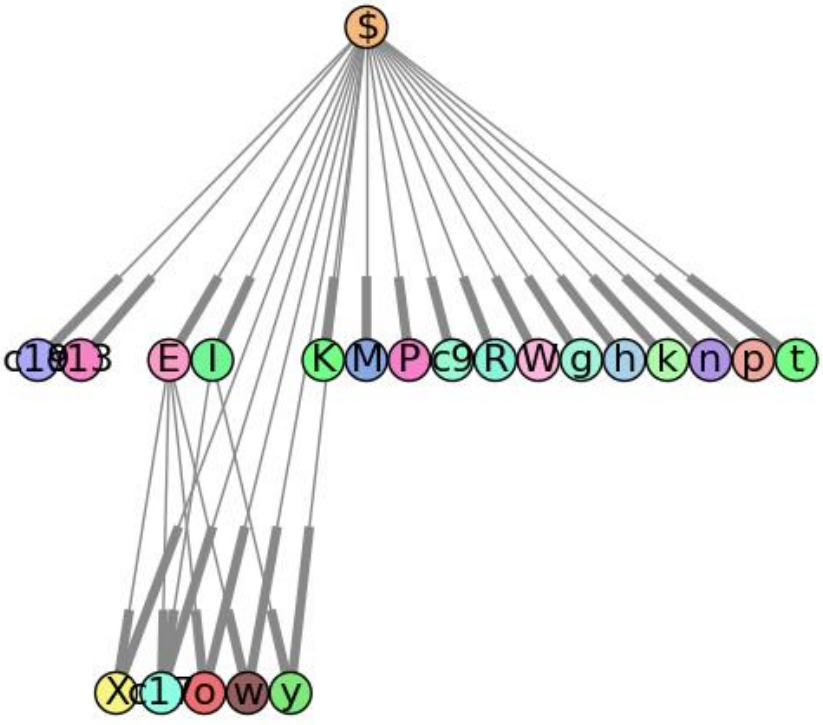



**Figure 25: Ancestry graph after removal of scars with high probability. As the amount of scar sites is very finite, the probability of multiple rare scars occurring in the same lineage branch is very low, resulting in a lineage tree which in its longest branches has 2 rare scars.**

## Conclusion

The reads derived from scartrace samples contain errors due to sequencing and PCR. The sequencing errors can be corrected for by collapsing the reads by combining a Hamming graph and the average base-calling confidence scores responsible for a substitution. An algorithm to quickly approximate the required Hamming graph was designed and implemented making the algorithm's time-complexity feasible. The drawback of collapsing based on Hamming distance is that indel-errors are not corrected for. As the currently performed amplicon simulations did not yield datasets which match the scar-data, a better, preferably in-vivo validation is required to test the capabilities of the collapser. The best experiment would be to sequence a sample with known concentrations of scar-sequences, the performance is than bases in how well these concentrations are reconstruction by using the collapser.

The scartrace data is not directly compatible with existing lineage tracing-methods due to scars occurring multiple times. This is solved by removing all scars which are likely to be introduced multiple times. By removal of abundant scars the scar data is transformed to be very similar to mutation based data and seems solvable for existing lineage-tracing algorithm. The driver mutation required by the E-K framework is artificially introduced. By using the existing E-K framework it is possible to generate a preliminary ancestry tree and lineage tree of the fin of a zebrafish. It would be very interesting to verify whether the reconstruction of the 2 trees from regenerated fins in the original scartrace experiment match with the original fin. The E-K method used here is the *error-free* method, meaning that there is no effort in mitigating sampling effects. The effect of not taking these errors into account was not studied, but should be checked for.

In the methodology used the amount of scar sites limits the depth of the inferred lineage tree. Highly probable scars populate many scar sites, these scars convey no information about the cell-lineages but consume sites, limiting the depth of the lineage tree reconstruction. Currently *somewhat* probable scars are also removed, which will carry some lineage information. A statistical method to find whether a scar was introduced multiple times in one experiment might help in shifting this threshold in an effort to keep more scars for lineage tracing.

Code for the read collapser is available at http://buysdb.nl/projects/spaghetto/source/

Special thanks to my supervisors and:
Tom Mokveld, for discussions about the Hamming graph approximation
Anna Alemany, for discussions about scartrace

References


1. Sulston, J. E., Schierenberg, E., White, J. G. & Thomson, J. N. The embryonic cell lineage of the nematode Caenorhabditis elegans. *Dev. Biol.* **100,** 64–119 (1983).

2. Lemischka, I. R., Raulet, D. H. & Mulligan, R. C. Developmental potential and dynamic behavior of hematopoietic stem cells. *Cell* **45,** 917–927 (1986).

3. Junker, J. P. *et al.* Massively parallel whole-organism lineage tracing using CRISPR / Cas9 induced genetic scars. *biorxiv* (2016). doi:10.1101/056499

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

5. McKenna, A. *et al.* Whole organism lineage tracing by combinatorial and cumulative genome editing. *Cell* **4,** 052712 (2016).

6. Schirmer, M. *et al.* Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Res.* **43,** 1–16 (2015).

7. Nakamura, K. *et al.* Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Res.* **39,** e90 (2011).

8. Alic, A. S., Ruzafa, D., Dopazo, J. & Blanquer, I. stand-alone error correction methods for NGS data. **6,** (2016).

9. Bastian, M., Heymann, S. & Jacomy, M. Gephi: An open source software for exploring and manipulating networks. BT - International AAAI Conference on Weblogs and Social. 361–362 (2009).

10. Jacomy, M., Venturini, T., Heymann, S. & Bastian, M. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PLoS One* **9,** 1–12 (2014).

11. Johnson, D. B. Efficient Algorithms for Shortest Paths in Sparse Networks. *J. ACM* **24,** 1–13 (1977).

12. Huang, W., Li, L., Myers, J. R. & Marth, G. T. ART: a next-generation sequencing read simulator. **28,** 593–594 (2012).

13. Reconstruction, P. Phylogenetic Trees - Parsimony. *Small* 1–7

14. Day, W. H. E., Johnson, D. S. & Sankoff, D. The computational complexity of inferring rooted phylogenies by parsimony. *Math. Biosci.* **81,** 33–42 (1986).

15. Felsenstein, J. Evolutionary trees from DNA sequences: A maximum likelihood approach. *J. Mol. Evol.* **17,** 368–376 (1981).

16. Kolaczkowski, B. & Thornton, J. W. Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature* **431,** 980–984 (2004).

17. Saitou, N. & Nei, M. The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees. *Mol. Biol. Evol.* **4,** 406–425 (1987).

18. Mihaescu, R., Levy, D. & Pachter, L. Why Neighbor-Joining Works. *Algorithmica* **54,** 1–24 (2009).

19. El-kebir, M., Oesper, L., Acheson-field, H. & Raphael, B. J. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. 62–70 (2015). doi:10.1093/bioinformatics/btv261

20. Stratton, M., Campbell, P. & Futreal, A. The cancer genome. *Nature* **458,** 719–724 (2009).

# Supplement

## Cas-9 lineage tracing mechanics

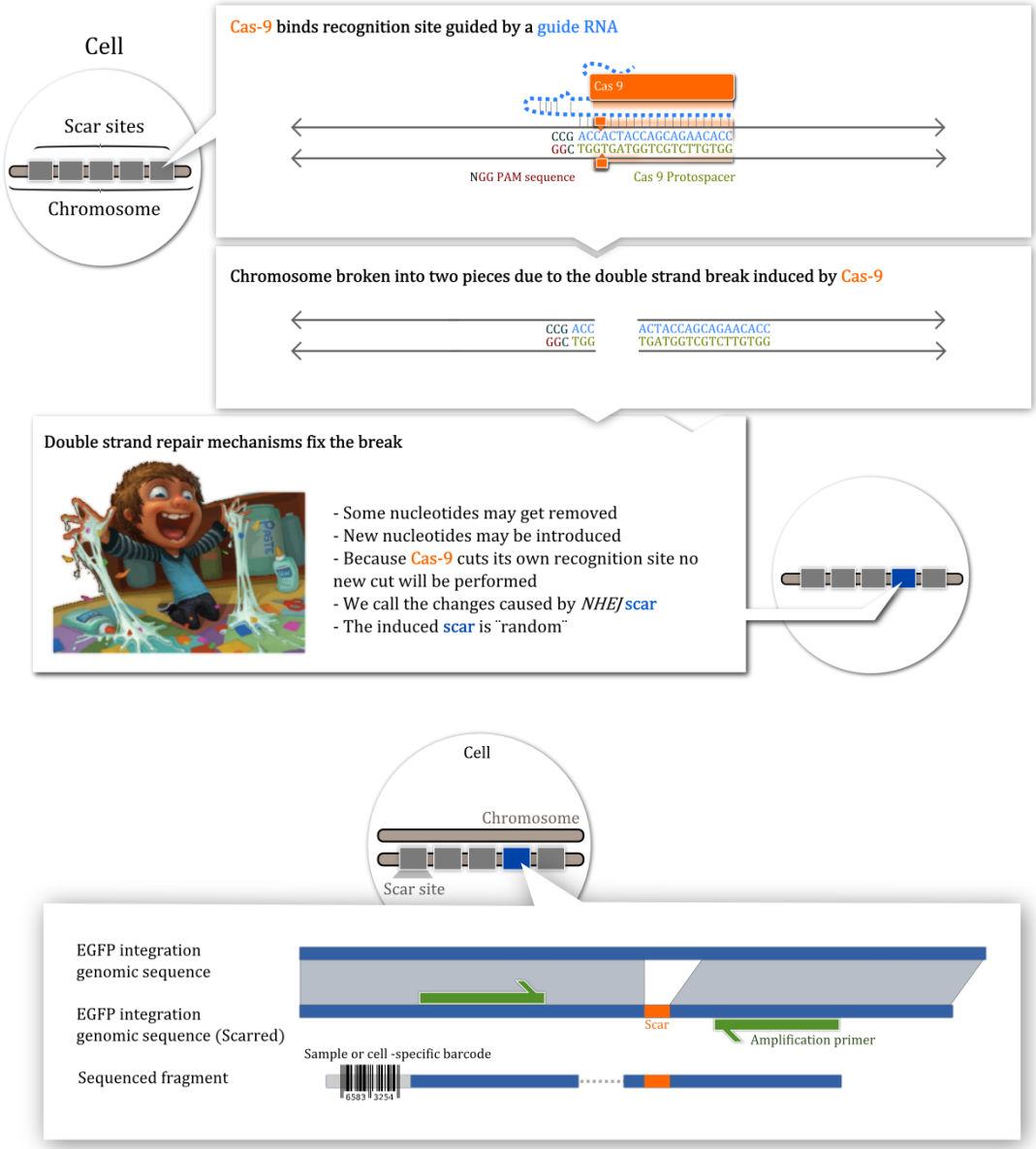



**Figure 26: Overview of the scartrace lineage marker storage mechanism. Scars are introduced by a cut of Cas-9 and the effects of NHEJ. The scar-site state is extracted by PCR and measured by sequencing.**

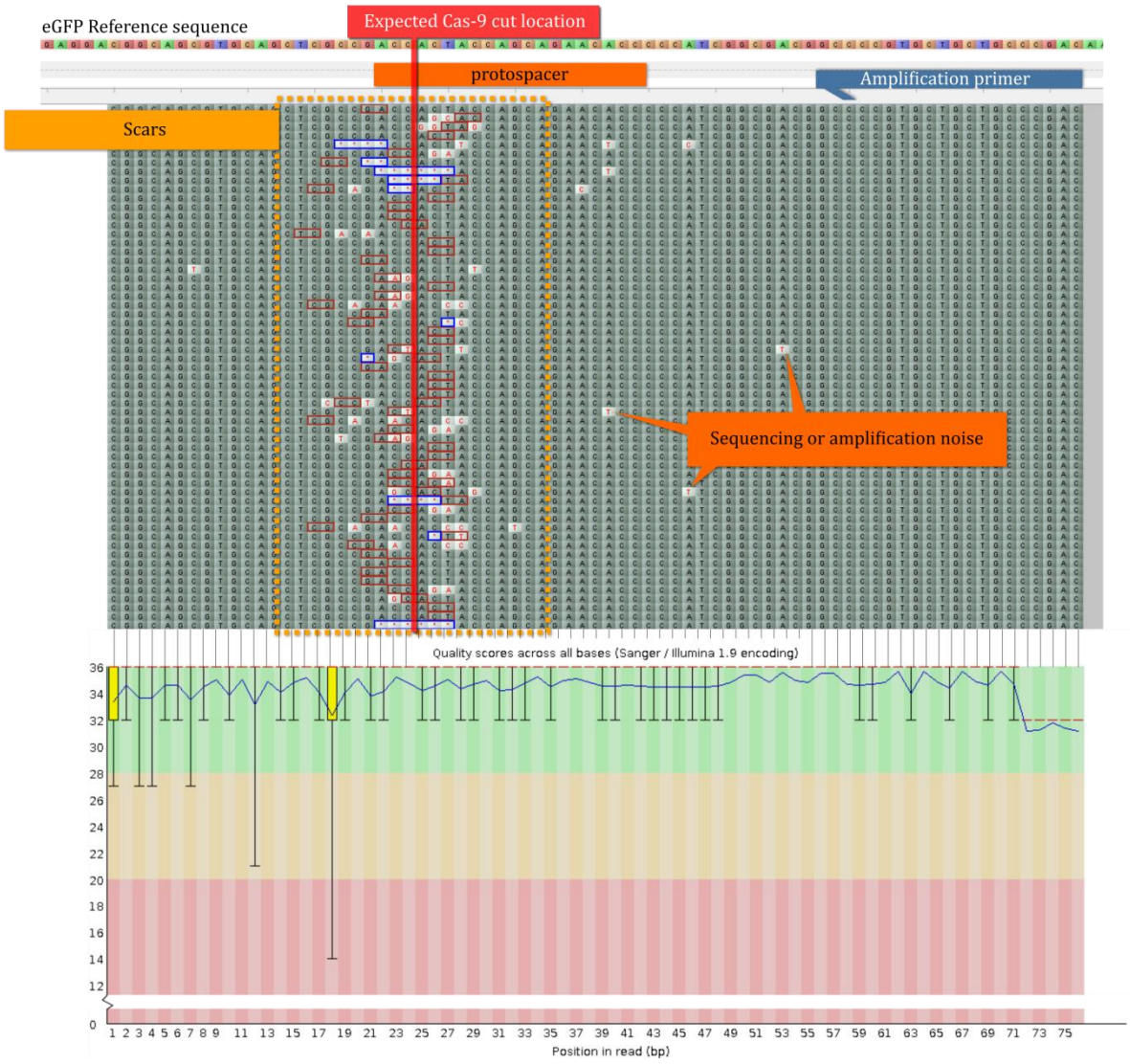



**Figure 27: Scartrace reads aligned to the eGFP reference sequence. Scars are located in the dashed orange box. Sequencing noise is visible as mismatching base-calls far away from the expected Cas-9 cutting location.**


Milne I, Stephen G, Bayer M, Cock PJA, Pritchard L, Cardle L, Shaw PD and Marshall D. 2013. Using Tablet for visual exploration of second-generation sequencing data. *Briefings in Bioinformatics* **14(2)**, 193-202.

## Collapsing simulated graph into expected lineage

Given the lineage graph **L**

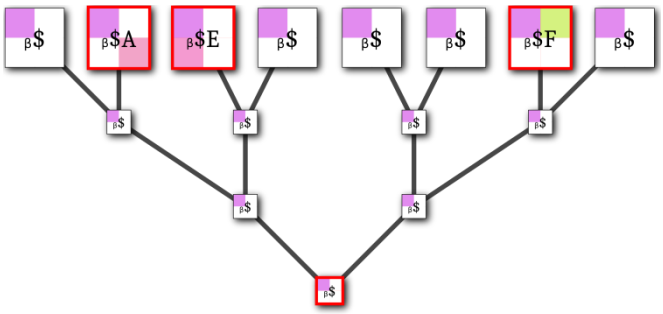



Figure 28: Simulated lineage graph. Red stroked cells introduce a new lineage branch.

for every non-branch identifying cell **n** with branch id **i**:

> find the branch **i** founder node **f**
> take outgoing edges of **n**,
> with branch id ≠ **i** , call it **o**
> add edge from **f** to all **o**
> remove **n** from **L**

resulting L is the expected lineage graph

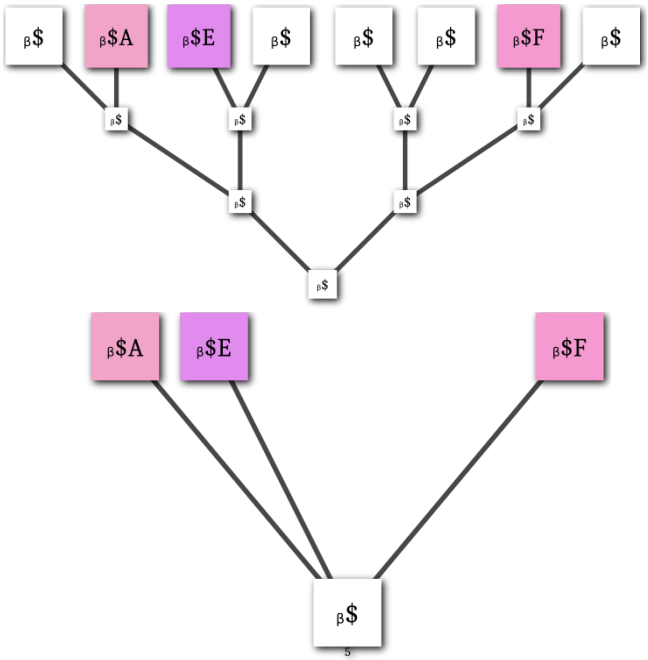



Figure 29: Expected lineage graph. This is the best reconstruction possible given the lineage-markers in L.

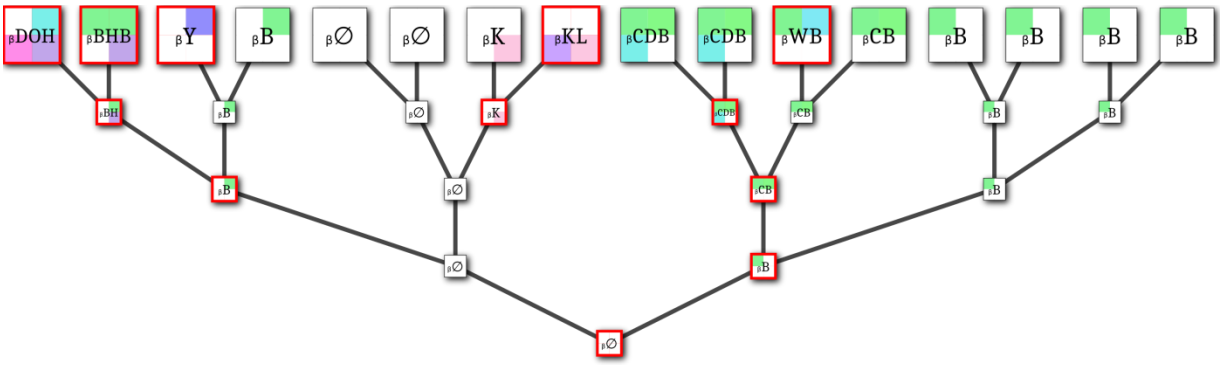



Figure 30: Simulated lineage tree with scar B introduced multiple times.

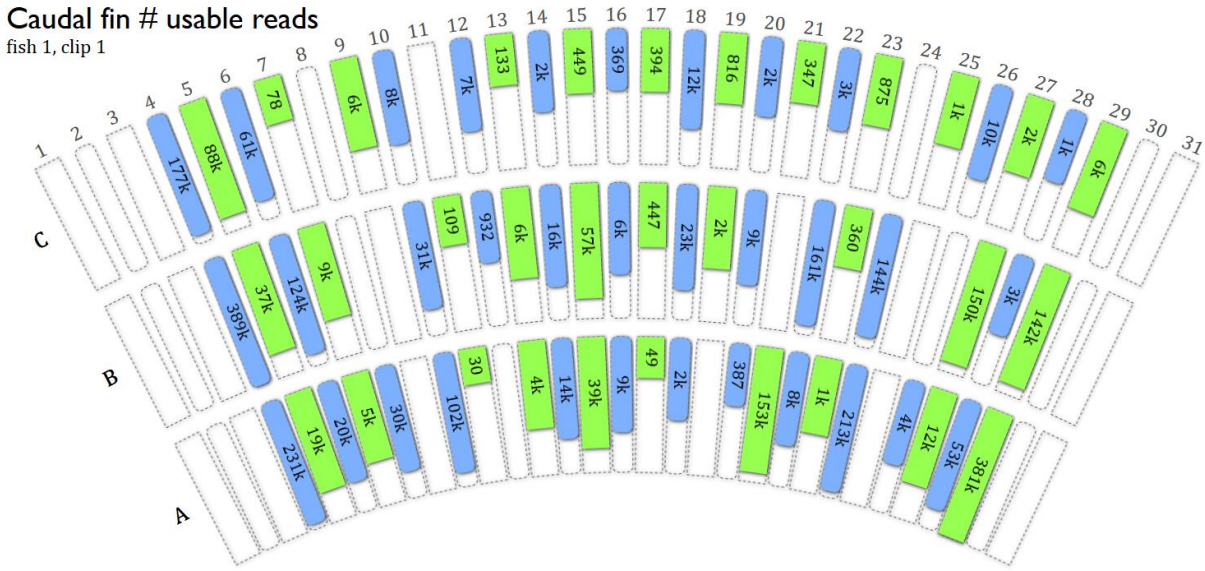
## Fin data analysis

All reads not containing the exact primer sequence are discarded. The reads were de-multiplexed by checking for an exact match with the sample-barcodes.

**Final FASTQ statistics table** ./data/fin/step=01_name=failedDemultiplexing_run=demultiplexed_flags=failedBarcode,R1.fq... 83 more

| sample | barcode | name | flags | #reads | #singletons | #unique | avg abundance | avg len | pctMapped | mapped | unmapped | #N | avg confidence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11-A-i-bc63-CTATGATC-f1-c1 | CTATGATC | scars | unfiltered,dmx | 301 | 55 | 69 | 4.36 | 76.0 | 100.0 | 301 | 0 | 0 | |
| 11-B-i-bc57-AGCGTCGA-f1-c1 | AGCGTCGA | scars | unfiltered,dmx | 438 | 58 | 79 | 5.54 | 76.0 | 100.0 | 438 | 0 | 0 | |
| 11-C-i-bc94-CACTGATC-f1-c1 | CACTGATC | scars | unfiltered,dmx | 191 | 41 | 50 | 3.82 | 76.0 | 100.0 | 191 | 0 | 0 | |
| 12-B-bone-bc80-GTACGATC-f1-c1 | GTACGATC | scars | unfiltered,dmx | 5539 | 386 | 491 | 11.28 | 76.0 | 99.93 | 5535 | 4 | 0 | |
| 13-A-i-bc81-ACTGATCG-f1-c1 | ACTGATCG | scars | unfiltered,dmx | 6899 | 454 | 556 | 12.41 | 76.0 | 99.1 | 6837 | 62 | 13 | |
| 12-C-bone-bc27-GACACGAG-f1-c1 | GACACGAG | scars | unfiltered,dmx | 22502 | 1448 | 1827 | 12.32 | 76.0 | 99.92 | 22485 | 17 | 2 | |
| 13-C-i-bc93-TGTACATC-f1-c1 | TGTACATC | scars | unfiltered,dmx | 1441 | 140 | 176 | 8.19 | 76.0 | 99.93 | 1440 | 1 | 0 | |
| 10-C-bone-bc26-CGTGTGAG-f1-c1 | CGTGTGAG | scars | unfiltered,dmx | 27632 | 1496 | 1905 | 14.5 | 76.0 | 99.96 | 27622 | 10 | 5 | |
| 13-B-i-bc17-TCACACGC-f1-c1 | TCACACGC | scars | unfiltered,dmx | 26745 | 1648 | 2050 | 13.05 | 76.0 | 99.96 | 26734 | 11 | 0 | |
| 14-C-bone-bc92-ACAGTCGA-f1-c1 | ACAGTCGA | scars | unfiltered,dmx | 9131 | 673 | 844 | 10.82 | 76.0 | 99.93 | 9125 | 6 | 2 | |
| | | failedDemultiplexing | failedLength,R2 | 95197 | 71375 | 72953 | 1.3 | 74.8514869166 | 33.31 | 31712 | 63485 | 704059 | |
| 14-A-bone-bc64-GCTCAATC-f1-c1 | GCTCAATC | scars | unfiltered,dmx | 73423 | 3589 | 4410 | 16.65 | 76.0 | 99.96 | 73391 | 32 | 10 | |
| 15-C-i-bc91-GTGCACGA-f1-c1 | GTGCACGA | scars | unfiltered,dmx | 2333 | 257 | 311 | 7.5 | 76.0 | 99.96 | 2332 | 1 | 0 | |
| 10-B-bone-bc55-TCCGAGAT-f1-c1 | TCCGAGAT | scars | unfiltered,dmx | 106419 | 5163 | 6389 | 16.66 | 76.0 | 99.95 | 106363 | 56 | 6 | |
| 14-B-bone-bc83-CAGTCTCG-f1-c1 | CAGTCTCG | scars | unfiltered,dmx | 83044 | 4149 | 5030 | 16.51 | 76.0 | 99.94 | 82995 | 49 | 3 | |
| 16-C-bone-bc90-CACTGCGA-f1-c1 | CACTGCGA | scars | unfiltered,dmx | 1413 | 169 | 215 | 6.57 | 76.0 | 99.93 | 1412 | 1 | 0 | |
| 17-A-i-bc65-AGCGTTCG-f1-c1 | AGCGTTCG | scars | unfiltered,dmx | 1175 | 133 | 174 | 6.75 | 76.0 | 99.83 | 1173 | 2 | 0 | |
| 17-B-i-bc81-ACTGATCG-f1-c1 | ACTGATCG | scars | unfiltered,dmx | 2445 | 214 | 270 | 9.06 | 76.0 | 99.96 | 2444 | 1 | 0 | |
| | | failedDemultiplexing | failedBarcode,R1 | 131077 | 58037 | 61347 | 2.14 | 75.7280072019 | 69.3 | 90838 | 40239 | 4093 | |
| 17-C-i-bc89-TGTACCGA-f1-c1 | TGTACCGA | scars | unfiltered,dmx | 1876 | 171 | 222 | 8.45 | 76.0 | 100.0 | 1876 | 0 | 0 | |
| 18-A-bone-bc66-TAGACTCG-f1-c1 | TAGACTCG | scars | unfiltered,dmx | 7086 | 515 | 644 | 11.0 | 76.0 | 99.9 | 7079 | 7 | 0 | |
| 16-B-bone-bc77-ACTGAATC-f1-c1 | ACTGAATC | scars | unfiltered,dmx | 36225 | 1947 | 2376 | 15.25 | 76.0 | 99.97 | 36213 | 12 | 3 | |
| 19-A-i-bc67-CTATGTCG-f1-c1 | CTATGTCG | scars | unfiltered,dmx | 117 | 22 | 30 | 3.9 | 76.0 | 100.0 | 117 | 0 | 0 | |
| 16-A-bone-bc14-GTCGTGAA-f1-c1 | GTCGTGAA | scars | unfiltered,dmx | 41668 | 1840 | 2263 | 18.41 | 76.0 | 99.95 | 41649 | 19 | 0 | |
| 19-B-i-bc76-GTACGCGA-f1-c1 | GTACGCGA | scars | unfiltered,dmx | 1596 | 151 | 192 | 8.31 | 76.0 | 100.0 | 1596 | 0 | 0 | |
| 19-B-i-bc19-GACACCGC-f1-c1 | GACACCGC | scars | unfiltered,dmx | 5265 | 494 | 632 | 8.33 | 76.0 | 99.96 | 5263 | 2 | 0 | |
| 19-C-i-bc28-ATGTGGAG-f1-c1 | ATGTGGAG | scars | unfiltered,dmx | 4027 | 304 | 414 | 9.73 | 76.0 | 99.9 | 4023 | 4 | 0 | |
| 18-C-bone-bc77-ACTGAATC-f1-c1 | ACTGAATC | scars | unfiltered,dmx | 28778 | 1415 | 1768 | 16.28 | 76.0 | 99.9 | 28750 | 28 | 15 | |
| 20-A-bone-bc68-GCTCATCG-f1-c1 | GCTCATCG | scars | unfiltered,dmx | 933 | 121 | 155 | 6.02 | 76.0 | 99.79 | 931 | 2 | 0 | |
| 18-B-bone-bc18-CGTGTCGC-f1-c1 | CGTGTCGC | scars | unfiltered,dmx | 51746 | 3053 | 3846 | 13.45 | 76.0 | 99.65 | 51564 | 182 | 5 | |
| 20-C-bone-bc88-GTACGGAT-f1-c1 | GTACGGAT | scars | unfiltered,dmx | 8422 | 553 | 707 | 11.91 | 76.0 | 99.96 | 8419 | 3 | 0 | |
| 20-B-bone-bc78-TGCATATC-f1-c1 | TGCATATC | scars | unfiltered,dmx | 28353 | 1787 | 2194 | 12.92 | 76.0 | 99.9 | 28326 | 27 | 1 | |
| 21-B-i-bc75-CAGTCCGA-f1-c1 | CAGTCCGA | scars | unfiltered,dmx | 119 | 27 | 34 | 3.5 | 76.0 | 100.0 | 119 | 0 | 0 | |
| 21-C-i-bc87-CAGTCGAT-f1-c1 | CAGTCGAT | scars | unfiltered,dmx | 1813 | 203 | 255 | 7.11 | 76.0 | 99.89 | 1811 | 2 | 0 | |
| 15-A-i-bc13-CATCAGAA-f1-c1 | CATCAGAA | scars | unfiltered,dmx | 159912 | 6932 | 8441 | 18.94 | 76.0 | 99.94 | 159819 | 93 | 23 | |
| 22-A-bone-bc69-AGCGTGAT-f1-c1 | AGCGTGAT | scars | unfiltered,dmx | 33002 | 1813 | 2237 | 14.75 | 76.0 | 99.94 | 32982 | 20 | 4 | |
| 22-C-bone-bc74-TGCATCGA-f1-c1 | TGCATCGA | scars | unfiltered,dmx | 9037 | 461 | 590 | 15.32 | 76.0 | 99.94 | 9032 | 5 | 9 | |
| 23-A-i-bc70-TAGACGAT-f1-c1 | TAGACGAT | scars | unfiltered,dmx | 4888 | 395 | 512 | 9.55 | 76.0 | 99.88 | 4882 | 6 | 2 | |
| 23-B-i-bc71-CTATGGAT-f1-c1 | CTATGGAT | scars | unfiltered,dmx | 540 | 51 | 65 | 8.31 | 76.0 | 100.0 | 540 | 0 | 0 | |
| 23-C-i-bc29-CTAACCGC-f1-c1 | CTAACCGC | scars | unfiltered,dmx | 4877 | 424 | 526 | 9.27 | 76.0 | 99.9 | 4872 | 5 | 0 | |
| 1-X-i-bc50-ATTCGTCG-f1-c1 | ATTCGTCG | scars | unfiltered,dmx | 135431 | 5948 | 7295 | 18.56 | 76.0 | 99.94 | 135348 | 83 | 4 | |
| | | failedDemultiplexing | failedPrimer,rc,R2 | 336803 | 180055 | 189528 | 1.78 | 76.0 | 68.07 | 229257 | 107546 | 6217 | |
| 24-B-bone-bc20-ATGTGCGC-f1-c1 | ATGTGCGC | scars | unfiltered,dmx | 44690 | 2099 | 2573 | 17.37 | 76.0 | 99.95 | 44668 | 22 | 0 | |
| 25-B-i-bc79-CAGTCATC-f1-c1 | CAGTCATC | scars | unfiltered,dmx | 131 | 32 | 38 | 3.45 | 76.0 | 100.0 | 131 | 0 | 0 | |
| 25-C-i-bc84-GTACGTCG-f1-c1 | GTACGTCG | scars | unfiltered,dmx | 3378 | 336 | 415 | 8.14 | 76.0 | 99.94 | 3376 | 2 | 0 | |
| 26-A-bone-bc72-GCTCAGAT-f1-c1 | GCTCAGAT | scars | unfiltered,dmx | 23031 | 1210 | 1543 | 14.93 | 76.0 | 99.97 | 23024 | 7 | 3 | |
| 10-A-bone-bc12-TGATGTCC-f1-c1 | TGATGTCC | scars | unfiltered,dmx | 398811 | 13000 | 16142 | 24.71 | 76.0 | 99.95 | 398615 | 196 | 28 | |
| 15-B-i-bc82-TGCATTCG-f1-c1 | TGCATTCG | scars | unfiltered,dmx | 352495 | 8535 | 10460 | 33.7 | 76.0 | 99.96 | 352340 | 155 | 276 | |
| 26-C-bone-bc86-TGCATGAT-f1-c1 | TGCATGAT | scars | unfiltered,dmx | 42650 | 2203 | 2716 | 15.7 | 76.0 | 99.95 | 42628 | 22 | 0 | |
| 27-C-i-bc82-TGCATTCG-f1-c1 | TGCATTCG | scars | unfiltered,dmx | 3848 | 395 | 505 | 7.62 | 76.0 | 99.97 | 3847 | 1 | 0 | |
| 27-A-i-bc49-GAATCTCG-f1-c1 | GAATCTCG | scars | unfiltered,dmx | 73608 | 3210 | 3900 | 18.87 | 76.0 | 99.96 | 73579 | 29 | 5 | |
| 21-A-i-bc15-ACGACGAA-f1-c1 | ACGACGAA | scars | unfiltered,dmx | 288513 | 12617 | 15193 | 18.99 | 76.0 | 99.85 | 288082 | 431 | 9 | |
| 28-B-bone-bc74-TGCATCGA-f1-c1 | TGCATCGA | scars | unfiltered,dmx | 11301 | 813 | 1017 | 11.11 | 76.0 | 99.93 | 11293 | 8 | 0 | |
| 28-C-bone-bc85-ACTGAGAT-f1-c1 | ACTGAGAT | scars | unfiltered,dmx | 4426 | 398 | 493 | 8.98 | 76.0 | 99.95 | 4424 | 2 | 1 | |
| 29-C-i-bc30-GCTTGCGC-f1-c1 | GCTTGCGC | scars | unfiltered,dmx | 37436 | 1859 | 2315 | 16.17 | 76.0 | 99.97 | 37423 | 13 | 0 | |
| 28-A-bone-bc72-GCTCAGAT-f1-c1 | GCTCAGAT | scars | unfiltered,dmx | 327826 | 8779 | 10729 | 30.56 | 76.0 | 99.95 | 327676 | 150 | 199 | |
| 2-X-bone-bc32-TAGGTCGC-f1-c1 | TAGGTCGC | scars | unfiltered,dmx | 196732 | 6602 | 8153 | 24.13 | 76.0 | 99.96 | 196659 | 73 | 18 | |
| 22-B-bone-bc75-CAGTCCGA-f1-c1 | CAGTCCGA | scars | unfiltered,dmx | 735742 | 16947 | 20714 | 35.52 | 76.0 | 99.89 | 734899 | 843 | 535 | |
| 4-C-bone-bc23-GACACTCA-f1-c1 | GACACTCA | scars | unfiltered,dmx | 1420 | 144 | 195 | 7.28 | 76.0 | 99.86 | 1418 | 2 | 0 | |
| 27-B-i-bc79-CAGTCATC-f1-c1 | CAGTCATC | scars | unfiltered,dmx | 709535 | 17642 | 21681 | 32.73 | 76.0 | 99.94 | 709130 | 405 | 447 | |
| 5-A-i-bc58-TAGACCGA-f1-c1 | TAGACCGA | scars | unfiltered,dmx | 86246 | 3944 | 4873 | 17.7 | 76.0 | 99.95 | 86199 | 47 | 0 | |
| 29-B-i-bc22-CGTGTTCA-f1-c1 | CGTGTTCA | scars | unfiltered,dmx | 756291 | 20334 | 25138 | 30.09 | 76.0 | 99.96 | 756004 | 287 | 98 | |
| 4-A-bone-bc56-CGGATGAT-f1-c1 | CGGATGAT | scars | unfiltered,dmx | 468119 | 15773 | 19302 | 24.25 | 76.0 | 99.95 | 467875 | 244 | 45 | |
| 5-B-i-bc52-CGGATTCG-f1-c1 | CGGATTCG | scars | unfiltered,dmx | 150551 | 6291 | 7711 | 19.52 | 76.0 | 99.95 | 150470 | 81 | 14 | |
| 6-A-bone-bc59-CTATGCGA-f1-c1 | CTATGCGA | scars | unfiltered,dmx | 119383 | 4904 | 5961 | 20.03 | 76.0 | 99.95 | 119323 | 60 | 2 | |
| 7-A-i-bc60-GCTCACGA-f1-c1 | GCTCACGA | scars | unfiltered,dmx | 28197 | 1500 | 1855 | 15.2 | 76.0 | 99.97 | 28189 | 8 | 0 | |
| 7-B-i-bc16-TGATGGAA-f1-c1 | TGATGGAA | scars | unfiltered,dmx | 43321 | 2108 | 2593 | 16.71 | 76.0 | 99.94 | 43293 | 28 | 5 | |
| 7-C-i-bc96-ACAGTATC-f1-c1 | ACAGTATC | scars | unfiltered,dmx | 644 | 93 | 119 | 5.41 | 76.0 | 100.0 | 644 | 0 | 0 | |
| 5-C-i-bc73-ACTGACGA-f1-c1 | ACTGACGA | scars | unfiltered,dmx | 280908 | 10947 | 13479 | 20.85 | 76.0 | 99.93 | 280799 | 109 | 11 | |
| 8-B-bone-bc54-ATTCGGAT-f1-c1 | ATTCGGAT | scars | unfiltered,dmx | 150 | 21 | 28 | 5.36 | 76.0 | 100.0 | 150 | 0 | 0 | |
| 9-A-i-bc62-TAGACATC-f1-c1 | TAGACATC | scars | unfiltered,dmx | 383 | 59 | 81 | 4.73 | 76.0 | 99.74 | 382 | 1 | 0 | |
| 9-B-i-bc78-TGCATATC-f1-c1 | TGCATATC | scars | unfiltered,dmx | 112 | 18 | 26 | 4.31 | 76.0 | 100.0 | 112 | 0 | 0 | |
| 9-C-i-bc95-GTGCAATC-f1-c1 | GTGCAATC | scars | unfiltered,dmx | 32494 | 1860 | 2286 | 14.21 | 76.0 | 99.95 | 32477 | 17 | 10 | |
| 24-A-bone-bc73-ACTGACGA-f1-c1 | ACTGACGA | scars | unfiltered,dmx | 1314030 | 25408 | 31130 | 42.21 | 76.0 | 99.92 | 1312953 | 1077 | 664 | |
| 6-C-bone-bc25-TCACAGAG-f1-c1 | TCACAGAG | scars | unfiltered,dmx | 260206 | 10104 | 12360 | 21.05 | 76.0 | 99.93 | 260028 | 178 | 32 | |
| 8-A-bone-bc61-AGCGTATC-f1-c1 | AGCGTATC | scars | unfiltered,dmx | 151937 | 6166 | 7585 | 20.03 | 76.0 | 99.94 | 151850 | 87 | 9 | |
| 6-B-bone-bc53-GAATCGAT-f1-c1 | GAATCGAT | scars | unfiltered,dmx | 534209 | 18256 | 22223 | 24.04 | 76.0 | 99.93 | 533831 | 378 | 36 | |
| 29-A-i-bc76-GTACGCGA-f1-c1 | GTACGCGA | scars | unfiltered,dmx | 2102455 | 38334 | 46218 | 45.49 | 76.0 | 99.96 | 2101537 | 918 | 1315 | |

Figure 31: Read counts after demultiplexing




Filtering on alpha values to obtain input for the lineage tracing algorithm was performed by taking into account only samples with at least 500 reads. Scars are kept which are present at least in two samples and with an introduction probability of at most 0.0001.

### Examples of erroneous sequences, observed when sequencing Wt only

>Expected (Wt sequence)
GGACGGCAGCGTGCAGCTCGCCGACCACTACCAGCAGAACACCCCCATCGGCGACG

>NW15-a1-31
CGCACGCCCCCGGCCCGCCGCCGTCCACCACCCCCCCCACCCCCCCATCGGCGACG

>NW14-a1-687
CGTCCGGCGCGTGGACCCGCCCGCCCCACACCCCCCCCACCCCCCCATCGGCGACG

>NW15-a1-692
GTAAGTCAGCGTTCAGCTAGACGACCACTATCAGAAGATCACAACAATAGGAGAAG

>NW12-a1-700
GGCCGGCCGCGGGCACCCCGCCGACCCCTCCCAGCCGCACCCCCCCCCCGGCGACG

>NW12-a1-712
CGCACGGAAGCTTGCTGCCCGCCGACCACTCCCCCCCCACCACCCCATCGGCGACG

### Simulation of NHEJ scars

The scars are induced by a double strand break followed by non-homologous end joining. It is interesting to find whether it is possible to model the scar creation process in order to learn about the data. (The simulator is incorporated in the Hamming collapser tool).

NHEJ removes bases (resection) and tries to find microhomologies in order to re-join the two sequences separated by the DSB. The process is shown in Figure 32.

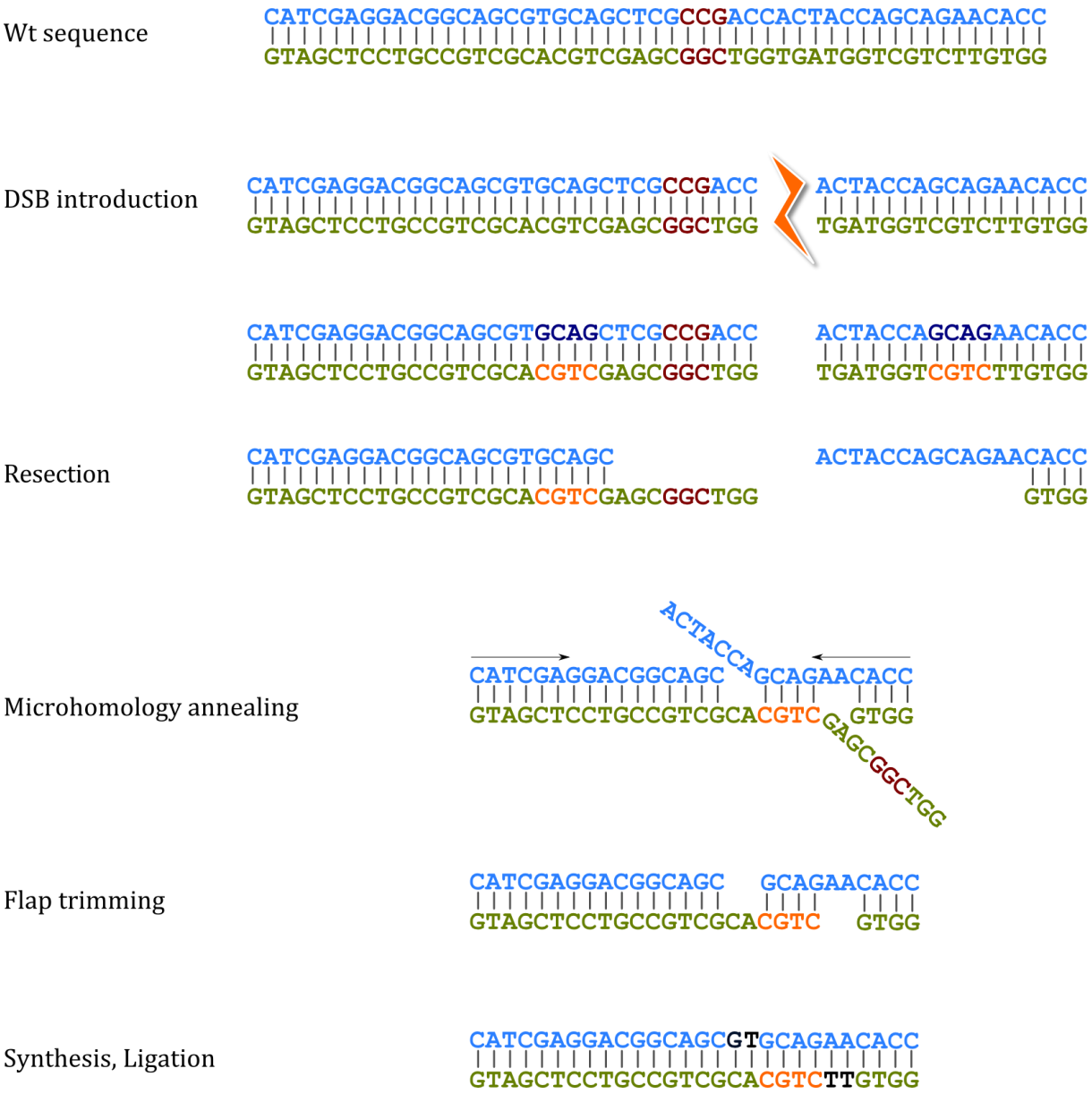



**Figure 32: Non-homologous end joining, schematically. When a DSB is introduced, non-homologous end joining starts repair by removing (damaged) nucleotides. A small overlap, a *microhomology* between the sequences is required in order to anneal the sequences. When the sequences are anealed the missing nucleotides are filled in. For the filling in, special translesion-polymerases are used, which sometimes introduce wrong or more bases, resulting in an (random) insert.**

Simulation starts with the sequences around the expected location of the DSB.

```
DSBRepairSim, INFO: Wildtype structure:
AGCTGGAGTACAACTACAACAGCCACAACGTCTATATCATGGCCGACAAGCAGAAGAACGGCATCAAGGTGAACTTCAAGATCCGCCACAACATCGAGGACGGCAGCGTGCAGCTCGCCGACCACTACCAGCAGAACACCCCCATCGGCGACGGCCCCGTGCTGCTGCCCGAC
TCGACCTCATGTTGATGTTGTCGGTGTTGCAGATATAGTACCGGCTGTTCGTCTTCTTGCCGTAGTTCCACTTGAAGTTCTAGGCGGTGTTGTAGCTCCTGCCGTCGCACGTCGAGCGGCTGGTGATGGTCGTCTTGTGGGGGTAGCCGCTGCCGGGGCACGACGACGGGCTG
DSBRepairSim, INFO: Flanking sequences:
AGCTGGAGTACAACTACAACAGCCACAACGTCTATATCATGGCCGACAAGCAGAAGAACGGCATCAAGGTGAACTTCAAGATCCGCCACAACATCGAGGACGGCAGCGTGCAGCTCGCCGACC
                                                                                                                                                                TGATGGTCGTCTTGTGGGGGTAGCCGCTGCCGGGGCACGACGACGGGCTG
```

Then the two sequences are iteratively shifted over each other in a range of 50bp. For every position the amount of matching bases is recorded. When more bases overlap than mismatch, the resulting fragment is recorded. A single base is removed from one of the ends of the reference fragment at a time, and the same process is repeated. The highest amount of bases matching, resulting in a specific scar is the output of the simulation.

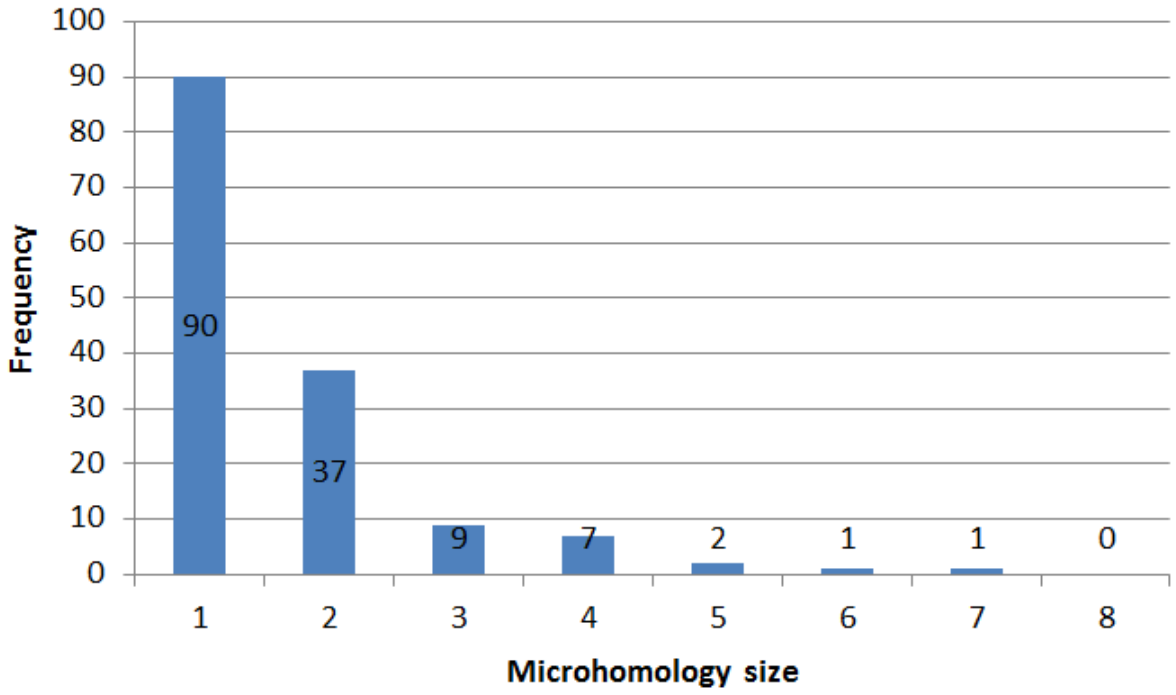
```
Sequence:CCGCCACAACATCGAGGACGGCAGCGTGCAGCTCGCCGACACCCCCATCGGCGACGGCCCCGTGCTGCTGCCCGAC
CIGAR:38M15D38M, Prior:0.00532428539159
Microhomology size:2 [2M 0MM], NWScore: 2.0
Nuclease steps:13 [1 12]
MicroHomologyAlignment:
AGCTGGAGTACAACTACAACAGCCACAACGTCTATATCATGGCCGACAAGCAGAAGAACGGCATCAAGGTGAACTTCAAGATCCGCCACAACATCGAGGACGGCAGCGTGCAGCTCGCCGAC
                                                                                                                                TGTGGGGGTAGCCGCTGCCGGGGCACGACGACGGGCTG


AC
||
AC
  Score=2

Sequence:CAAGCAGAAGAACGGCATCAAGGTGAACTACCAGCAGAACACCCCCATCGGCGACGGCCCCGTGCTGCTGCCCGAC
CIGAR:26S50M, Prior:0.0139187125205
Microhomology size:3 [3M 0MM], NWScore: 3.0
Nuclease steps:48 [48 0]
MicroHomologyAlignment:
AGCTGGAGTACAACTACAACAGCCACAACGTCTATATCATGGCCGACAAGCAGAAGAACGGCATCAAGGTGAACT
                                                                TGATGGTCGTCTTGTGGGGGTAGCCGCTGCCGGGGCACGACGACGGGCTG


ACT
|||
ACT
  Score=3
```

Using this I find 147 different scars, with microhomology sizes shown below. The NHEJ simulated scars match with many of the highly abundant scars observed in a scartrace dataset (Figure 33).



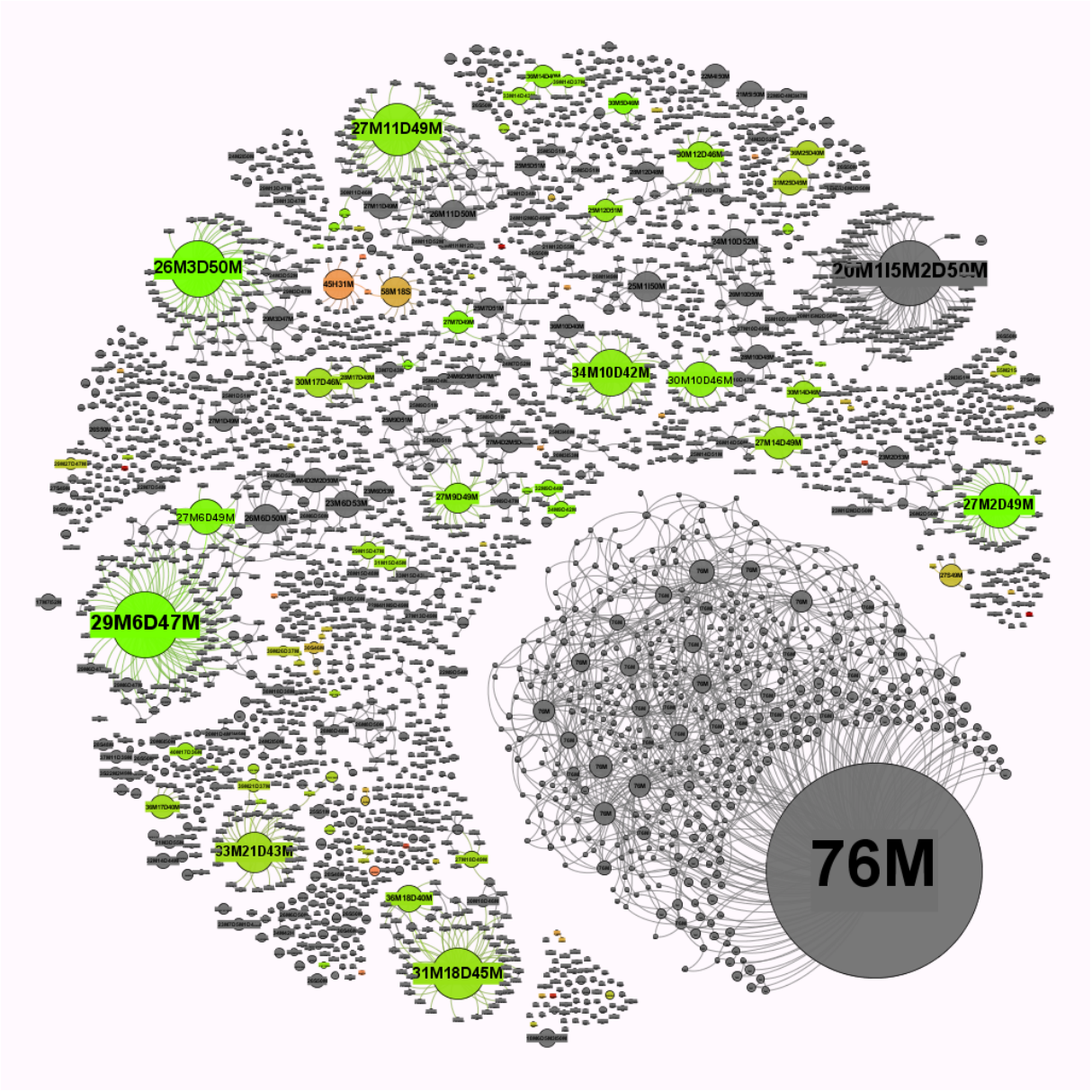



**Figure 33: A simple NHEJ simulation method was designed and the results are overlayed on observed scars. The 76M node reflects the reads from Wt scar sites. The colored nodes indicate that these sequences can be reached by NHEJ. The simulation does not simulate translesion-polymerase activity, therefore all nodes with an "I" in the alignment string are always grey. This simulation result strengthens the hypothesis that the sequences close in hamming distance to a scar are indeed sequencing errors, and not attraction basins of NHEJ.**