Ratatosk – Hybrid error correction of long reads enables accurate variant calling and assembly

Guillaume Holley[1], Doruk Beyter[1], Helga Ingimundardottir[1], Snædis Kristmundsdottir[1,2], Hannes P. Eggertsson[1], and Bjarni V. Halldorsson[1,2]

[1]deCODE genetics/Amgen Inc., Reykjavík, Iceland
[2]School of Technology, Reykjavik University, Reykjavík, Iceland

Abstract

Motivation: Long Read Sequencing (LRS) technologies are becoming essential to complement Short Read Sequencing (SRS) technologies for routine whole genome sequencing. LRS platforms produce DNA fragment reads, from 10^3 to 10^6 bases, allowing the resolution of numerous uncertainties left by SRS reads for genome reconstruction and analysis. In particular, LRS characterizes long and complex structural variants undetected by SRS due to short read length. Furthermore, assemblies produced with LRS reads are considerably more contiguous than with SRS while spanning previously inaccessible telomeric and centromeric regions. However, a major challenge to LRS reads adoption is their much higher error rate than SRS of up to 15%, introducing obstacles in downstream analysis pipelines.

Results: We present Ratatosk, a new error correction method for erroneous long reads based on a compacted and colored de Bruijn graph built from accurate short reads. Short and long reads color paths in the graph while vertices are annotated with candidate Single Nucleotide Polymorphisms. Long reads are subsequently anchored to the graph using exact and inexact k-mer matches to find paths corresponding to corrected sequences. We demonstrate that Ratatosk can reduce the raw error rate of Oxford Nanopore reads 6-fold on average with a median error rate as low as 0.28 %. Ratatosk corrected data maintain nearly 99 % accurate SNP calls and increase indel call accuracy by up to about 40 % compared to the raw data. An assembly of the Ashkenazi individual HG002 created from Ratatosk corrected Oxford Nanopore reads yields a contig N50 of 43.22 Mbp and outperforms high quality LRS assemblies using PacBio HiFi reads. In particular, the assembly of Ratatosk corrected reads contains about 2.5 times less errors than an assembly created from PacBio HiFi reads.

Availability: `https://github.com/DecodeGenetics/Ratatosk`

Contact: guillaume.holley@decode.is

1 Introduction

In Norse mythology, the squirrel Ratatöskr runs up and down the ash tree Yggdrasil, bearing envious words between the eagle at the top and the dragon at the bottom. Short read sequencing (SRS) has allowed for the accurate identification of small variants (SNPs and indels) in non-repetitive parts of the genome while long read sequencing (LRS) allows for the characterization of large and complex variations. We have designed Ratatosk to carry information between the two technologies with the hope of leveraging the benefits of both of them.

Oxford Nanopore Technologies (ONT) and Pacific Bioscience (PacBio) are LRS platforms [Logsdon et al., 2020] that produce long sequence reads ranging from 10^3 to 10^6 bases with an error rate up to 15% [Rang et al., 2018]. The high error rate of LRS reads is in part compensated by their lengths which increase their mapping accuracy, making LRS suitable for numerous applications in all fields of genomics. LRS used at high coverage on a few individuals [Audano et al., 2019] or low-medium coverage at population scale [Beyter et al., 2019] greatly improves the detection of Structural Variants (SVs) because the large size of ONT reads spans SVs breakpoints. Additionally, LRS reads can encompass large sections of highly repetitive regions in the human genome such as centromeres [Bzikadze and Pevzner, 2019], telomeres [Miga et al., 2020] and tandem repeats [Mitsuhashi et al., 2019]. Analyzing these regions with SRS is gruelling as the reads generally map ambiguously to multiple locations because of their limited size. Yet, centromeres play an important role in cancer genomics [Miga, 2019] while Short Tandem Repeat (STR) expansions associate with a number of genetic diseases [Kristmundsdottir et al., 2020]. LRS technologies have also enabled *de novo* haplotype-resolved assemblies with very few contig breaks [Porubsky et al., 2019]. Finally, LRS technologies overcome chemistry limitations of SRS, in particular GC bias [Chen et al., 2013] and PCR amplification artifacts [Kozarewa et al., 2009] causing uneven coverages for reads produced by Illumina platforms. Yet, the high error rate of LRS reads introduces algorithmic challenges in analyzing these data while filtering out the noise [Sedlazeck et al., 2018a]. Highly accurate LRS technologies [Wenger et al., 2019] that perform circular sequencing and generate highly accurate consensus sequences are emerging but the required resources are still prohibitive at a population scale. SRS data are therefore often used to complement to LRS data for SV breakpoint refinement [Sedlazeck et al., 2018b] and assembly polishing [Kolmogorov et al., 2019].

We present Ratatosk, a new method based on a compacted and colored de Bruijn graph for the hybrid correction of LRS reads using SRS data. Ratatosk is specifically designed to avoid over-correction with incorrect haplotypes or homologous regions as this would either remove true variants or add artificial ones. Ratatosk introduces several new features not included in other hybrid correction tools. First, SRS and LRS reads color vertices of the de Bruijn graph to highlight existing paths for the correction. Graph coloring enables pruning the search space when traversing the graph by removing chimeric paths. Second, LRS reads are anchored to the graph using both exact and inexact k-mer matches. The latter improves the anchoring of highly erroneous regions of the LRS reads. Third, the graph is annotated with candidate SNPs to disentangle

small variations between haplotypes that are difficult to capture from erroneous LRS reads. Finally, two passes of correction are performed using SRS and LRS reads separately to take advantage of all data available, as well as increasing k-mer sizes to remove errors made during the first correction pass.
The performance of LRS read error correction tools is usually evaluated by the error rate, genome coverage and different assembly metrics of the corrected reads [Marchet et al., 2020, Morisse et al., 2020] but there has been little investigation of the accuracy of variant calls on the corrected data. We demonstrate that Ratatosk can reduce the raw error rate of ONT reads 6-fold on average with a median error rate as low as 0.28 %. Ratatosk corrected data maintain nearly 99 % accurate SNP calls and substantially increase indel calls accuracy to nearly 92 % compared to the raw data. An assembly of the Ashkenazi individual HG002 [Zook et al., 2016] created from Ratatosk corrected ONT reads yields a contig N50 of 43.22 Mbp and outperforms two other high quality LRS assemblies on several metrics. In particular, the assembly of Ratatosk corrected reads contains about 2.5 times less misassemblies than an assembly created from PacBio HiFi reads.

1.1 Previous work

Methods for LRS reads correction belong to one of two categories: self-correction or hybrid correction. Self-correction methods refine the reads using information from the set of LRS reads alone while hybrid correction methods use information from a set of SRS reads originating from the same individuals. Overall, hybrid correction methods have been shown to outperform self-correction methods in terms of error rate and compute resource usage. However, a recurrent issue with most error correction methods is that they do not retain the phasing of the reads, hence limiting the usage of corrected data to mixed-haplotype assembly. We provide here a short overview of hybrid correction methods and refer to the reviews of Zhang et al., 2019, Fu et al., 2019 and Morisse et al., 2020 for more details about self-correction methods.

Most hybrid correction methods use a de Bruijn graph built from SRS reads as an index for the correction of LRS reads. The de Bruijn graph has been extensively used as a data structure for genome assembly [Pevzner et al., 2001, Idury and Waterman, 1995]. LoRDEC [Salmela and Rivals, 2014] builds a de Bruijn graph from SRS reads and anchors LRS reads to the graph. Subsequences that do not anchor to the graph are then corrected: paths which are similar to the uncorrected subsequences are extracted and used for correction. Jabba [Miclotte et al., 2016] is similar to LoRDEC besides that SRS reads are self-corrected before graph construction and LRS reads are anchored to the graph using Maximum Exact Matches to enable different k-mer lengths during correction. HG-CoLoR [Morisse et al., 2018] also uses self-corrected SRS reads and aligns them to the LRS reads to find overlaps. These overlaps anchor the reads onto a variable-order de Bruijn graph allowing for multiple k-mer lengths. Finally, FMLRC [Wang et al., 2018] indexes the de Bruijn graph using a multi-string Burrows-Wheeler Transform of the SRS reads. This representation is lightweight in memory, enables multiple k-mer lengths and stores implicitly k-mer frequencies. FMLRC has two passes of correction, one using a short k-mer and one using a long k-mer in order to simplify the graph for high complexity regions to correct. Unlike the above tools, CoLoRMap [Haghshenas et al., 2015]

constructs a weighted alignment graph from the mapping of the SRS reads to the LRS reads. The mapping provides paths in the graph that maximize the similarity with the subsequences to correct. CoLoRMap takes advantage of the paired-end information to leap over regions of LRS reads where no SRS reads map. We refer to the reviews of Zhang et al., 2019, Fu et al., 2019 and Morisse et al., 2020 for further information.

2 Results

We evaluated Ratatosk using our reference-guided preprocessing on a set of 4 Icelandic trios from deCODE genetics [Jonsson et al., 2017] and the Ashkenazi individual HG002 from Genome In A Bottle [Zook et al., 2016]. Each genome was sequenced with both Illumina and ONT platforms. Two HG002 data sets basecalled with Guppy 3.6 and Guppy 3.3 were employed as Guppy 3.6 significantly improves the raw read accuracy over Guppy 3.3. Genome coverage and N50 metrics are reported in Table 1 for the ONT reads. The short reads used are Illumina paired-end reads of length 151 bases with a mean coverage of 42x in the Icelandic trios and 61x in the HG002 data set. The Ratatosk corrected ONT reads were subsequently compared to the raw and FMLRC [Wang et al., 2018] corrected reads. The reviews of Zhang et al., 2019 and Fu et al., 2019 highlight FMLRC as one of the correction tools with the best overall performance among hybrid methods. Time and memory usage for Ratatosk and FMLRC are reported in Appendix, Section D. All ONT reads were subsequently aligned to the reference human genome GRCh38.p13 with minimap2 [Li, 2018] using the default ONT setting for further analysis.

Table 1: Genome coverage and N50 for the ONT reads of the child (C), father (F) and mother (M) in 4 Icelandic trios and the child in two HG002 data sets.

	Guppy version	Coverage			N50		
		C	F	M	C	F	M
T1	3.3	63.68	50.94	64.74	20,353	24,093	23,528
T2	3.3	55.68	67.95	70.46	25,767	22,496	20,439
T3	3.3	69.50	57.05	56.62	24,047	27,787	26,856
T4	3.3	57.07	57.40	64.28	23,111	15,234	26,634
HG002-1	3.6	46.72	NA	NA	52,311	NA	NA
HG002-2	3.3	108.10	NA	NA	22,260	NA	NA

2.1 Error rate

Tables 2a and 2b show the error rates for the ONT reads corrected by Ratatosk and FMLRC as well as the raw ONT reads. The mean error rate of the Ratatosk reads is about 1.9 times lower than the FMLRC reads and about 6.3 times lower than the raw reads. On the HG002 data sets, 50 % of the Ratatosk reads have an error rate of 0.28 % or below. This is up to 29 times lower than the raw reads and up to 3.5 times lower than the FMLRC reads. Details on the error rate calculations are given in Appendix, Section B.

Table 2: Error rates (in %) for the raw and corrected ONT reads in 4 Icelandic trios and two HG002 data sets. Best results are highlighted.

		Raw			FMLRC			Ratatosk		
		C	F	M	C	F	M	C	F	M
Mean	T1	11.89	11.20	10.89	3.85	3.56	3.33	**1.76**	**1.81**	**1.72**
	T2	10.52	11.20	10.14	3.20	3.48	2.94	**1.70**	**1.79**	**1.58**
	T3	9.98	10.52	10.78	3.07	3.16	3.47	**1.73**	**1.66**	**1.76**
	T4	10.74	11.19	10.18	3.26	3.57	2.94	**1.56**	**1.76**	**1.52**
Median	T1	9.95	9.10	8.84	1.41	1.18	1.11	**0.30**	**0.31**	**0.29**
	T2	8.37	9.05	8.22	1.00	1.15	0.88	**0.28**	**0.29**	**0.28**
	T3	7.94	8.42	8.72	1.03	0.99	1.27	**0.33**	**0.29**	**0.35**
	T4	8.91	9.34	8.16	1.23	1.33	0.96	**0.28**	**0.28**	**0.27**

(a) Error rates (in %) of the child (C), father (F) and mother (M) in 4 Icelandic trios.

	Raw		FMLRC		Ratatosk	
	Mean	Median	Mean	Median	Mean	Median
HG002-1	8.81	6.95	2.53	0.55	**1.48**	**0.27**
HG002-2	10.12	8.23	2.90	0.99	**1.59**	**0.28**

(b) Error rates (in %) for two HG002 data sets.

We also report in Figure 1 the number of supplementary alignments and the ratio of ambiguous bases as metrics for read quality. Supplementary alignments occur when an alignment cannot be represented as a single linear alignment [Li et al., 2009] but instead, as a set of linear alignments where the alignment with the greatest alignment score is selected as *primary* and the others as *supplementary*. The presence of supplementary alignments might indicate an SV large enough for the aligner to abandon mapping the read with a single linear alignment. Supplementary alignments might also indicate that the read has been partially over-corrected. Finally, ambiguous bases are bases from reads which do not align in the extremities of primary alignments (*soft-clipping*) but do align in at least one distant supplementary alignment of the same reads. The ratio of ambiguous bases measure the proportion of read bases mapping ambiguously because of chimeric reads [Marijon et al., 2020] or over-correction. More details are given in Appendix, Section C.

As shown in Figure 1, Ratatosk slightly decreases on average the number of supplementary alignments and the ratio of ambiguous bases compared to the raw reads. On the other hand, FMLRC increases the number of supplementary alignments by a factor 8.28 and increases the ratio of ambiguous bases by a factor 1.44. This suggests that Ratatosk can correct soft-clipped bases and chimeric reads while FMLRC is prone to over-correction.

2.2 Variant calling

There is a limited number of tools that can perform small variant calling on corrected LRS reads. Clair [Luo et al., 2020] and DeepVariant [Poplin et al., 2018] are machine learning based and can train a model given a training set of input reads. We use Clair for our evaluations as DeepVariant was several

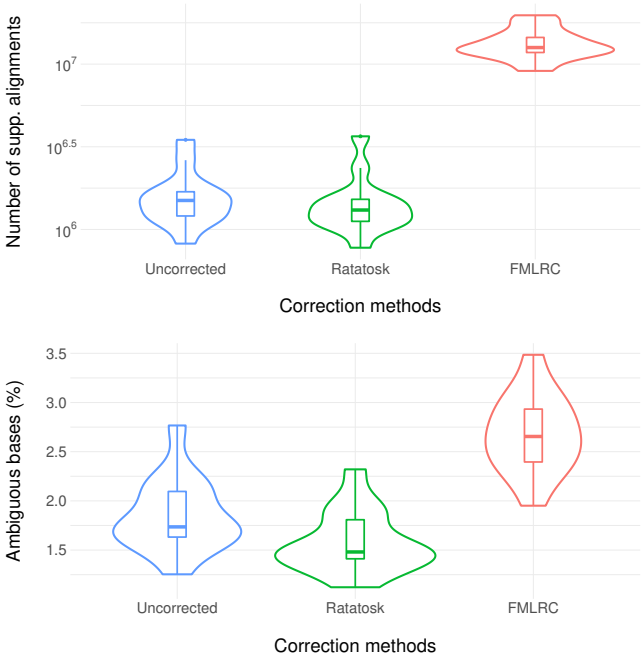


Figure 1: Number of supplementary alignments and ratio of ambiguous bases for the Icelandic trios and the HG002 data sets.

times slower to train on the raw ONT reads. Longshot [Edge and Bansal, 2019] was not used as it does not call indels while Medaka [Oxford Nanopore Technologies, accessed June 10th 2020, 2019] uses an error model specific to the raw ONT reads and hence, could not be applied to corrected data. A model was trained with Clair on the raw, Ratatosk and FMLRC ONT reads from HG002-1 using the truth set of variants less than 50 bases long in the high confidence regions (see Appendix, Section F). The different models generated for each type of input long reads were then used to call small variants on all genomes and variant calls were subsequently evaluated using rtg-tools [Krusche et al., 2019].

Given a variant truth set, rtg-tools automatically computes an optimal quality score threshold for the variant calls. Table 3a shows the variant calls accuracy for HG002-1 for which low quality variants below the optimal threshold are filtered out. On the other hand, Table 3b illustrates a standard setting for which all variants with the `FILTER` field set to `PASS` in the VCF files are used. With quality score filtering, SNP calls are nearly 99 % accurate for the raw and Ratatosk reads with a slight accuracy decrease in the SNPs called from the FMLRC reads. This demonstrates that SNPs are accurately represented in the raw reads and Ratatosk captures well the SNP candidates in the correction. However, indels are poorly represented in the raw reads and Ratatosk increases the indel calls accuracy by up to 40 % compared to the raw reads. When no filtering is applied, the difference of indel calls accuracy between raw and corrected reads is staggering. Indeed, the indel calls accuracy of raw reads shrinks to 19 % because a larger number of false positive indels are called compared to

Table 3: Small variant calls accuracy (in %) for the ONT reads from HG002-1 in high confidence (HC) and high confidence non repetitive (HCNR) regions. Best results are highlighted.

		SNPs			Indels		
		Precision	Recall	F1	Precision	Recall	F1
HCNR	Raw	**98.94**	98.66	**98.80**	76.41	83.72	79.90
	FMLRC	97.84	97.18	97.51	91.22	94.36	92.76
	Ratatosk	98.53	**99.05**	98.79	**94.90**	**97.34**	**96.10**
HC	Raw	**98.65**	97.98	98.31	74.47	40.01	52.05
	FMLRC	97.44	97.27	97.36	93.08	81.07	87.01
	Ratatosk	98.24	**99.17**	**98.70**	**93.84**	**89.81**	**91.78**

(a) Variants with quality scores below a threshold automatically computed by rtg-tools are filtered out.

		SNPs			Indels		
		Precision	Recall	F1	Precision	Recall	F1
HCNR	Raw	89.00	99.86	94.12	11.90	95.09	21.16
	FMLRC	77.97	99.80	87.55	51.83	98.85	68.01
	Ratatosk	**92.87**	**99.89**	**96.25**	**88.74**	**99.15**	**93.65**
HC	Raw	87.40	99.75	93.17	10.85	73.64	18.92
	FMLRC	77.91	99.77	87.50	59.49	93.78	72.80
	Ratatosk	**92.73**	**99.88**	**96.17**	**86.94**	**96.09**	**91.29**

(b) All variants with the `FILTER` field set to `PASS` in the VCF files are used.

the filtered calls. Indel call accuracy from the FMLRC reads decreases to 73 % while indels called from the Ratatosk reads decline only to 91.29 % accuracy, a 0.5 % reduction compared to the filtered indel calls.

No variant truth set is available for the Icelandic trios so Mendelian inheritance concordance was measured by rtg-tools instead, as shown in Table 4. Note that because the trios were basecalled with Guppy 3.3, another model was trained using HG002-2 for the raw reads and FMLRC corrected reads. Overall, small variants calls from Ratatosk reads are the most consistent with the calls from each parents or both parents across all trios.

2.3 Assembly

The Ratatosk corrected reads of HG002-1 were assembled using Flye 2.7.1 [Kolmogorov et al., 2019] and the produced contigs were evaluated with QUAST 5.0.2 [Gurevich et al., 2013] and Merqury [Rhie et al., 2020]. We compared the Flye assembly of Ratatosk corrected reads to a recent assembly made from PacBio HiFi reads with HiCanu [Nurk et al., 2020] and the reference assembly Ash1 v1.7 [Shumate et al., 2020] made from Illumina, ONT and PacBio HiFi reads assembled with MaSuRCA [Zimin et al., 2013]. Misassemblies reported by QUAST were filtered to exclude errors in known SV and segmental duplication sites as well as centromeric regions using a script from HELEN [Shafin et al., 2020]. The quality value represents a log-scaled probability of error for

Table 4: Mendelian concordance (in %) of small variants called on the ONT reads of 4 children from Icelandic trios with respect to the variant calls from their father (F), mother (M) and both parents (F+M). All variants with the `FILTER` field set to `PASS` in the VCF files are used by rtg-tools. Best results are highlighted.

	Raw			FMLRC			Ratatosk		
	F	M	F+M	F	M	F+M	F	M	F+M
T1	98.87	98.92	93.04	98.89	98.96	95.62	**99.49**	**99.48**	**98.03**
T2	98.90	98.78	93.05	99.01	99.00	95.89	**99.50**	**99.49**	**97.70**
T3	98.86	**98.92**	93.70	98.67	98.57	95.23	**98.94**	98.52	**96.25**
T4	98.80	98.98	93.39	98.95	99.12	96.02	**99.51**	**99.54**	**98.21**

the consensus basecalls while the k-mer completeness measures the proportion of k-mers shared between the assembly and an accurate SRS data set from the same individual.

Table 5: HG002-1 assembly metrics for Ratatosk corrected ONT reads assembled with Flye, PacBio HiFi reads assembled with HiCanu and the Ash1 reference assembly. Misassemblies are filtered to exclude errors in known SV and segmental duplication sites as well as centromeric regions. All metrics are computed by QUAST except k-mer completeness and quality value which are computed by Merqury. Best results are highlighted.

	Ratatosk + Flye	HiFi + HiCanu	Ash1
Reference coverage (%)	96.14	98.13	**98.50**
k-mer completeness (%)	97.50	**99.55**	97.67
Quality value	**47.73**	43.76	41.341
NG50 (Mbp)	43.22	**46.52**	33.65
NGA50 (Mbp)	**23.92**	20.82	15.62
# contigs	**807**	30,532	2,412
# misassemblies	**88**	237	187
# mismatches per 100 kbp	**118.97**	367.58	161.09
# indels per 100 kbp	27.27	67.15	**27**

As shown in Table 5, the Flye assembly of the Ratatosk reads is competitive with other high quality LRS assemblies and outperforms them on several metrics. In particular, the Ratatosk/Flye assembly has the best quality value, the largest NGA50 and the lowest number of contigs. Furthermore, the produced contigs contain only 88 misassemblies, about 2.1 times less than the Ash1 reference assembly and 2.7 times less than the HiFi/HiCanu assembly. The only metric where the Ratatosk/Flye assembly is outperformed by the other two is the reference genome coverage which is 2.36 % lower than in the Ash1 assembly. However, 1.96 % of the Ash1 assembly is derived from the reference genome GRCh38 while no post-processing was applied to the Ratatosk/Flye assembly. Overall, these results demonstrate that the correction performed by

Ratatosk is suited for producing highly contiguous assemblies of quality with very few errors.

3 Conclusion

We present Ratatosk, a hybrid error correction tool for long and erroneous reads designed for accurate variant calling and assembly. Ratatosk uses short and long reads to color paths in a compacted de Bruijn graph index and annotate vertices with candidate Single Nucleotide Polymorphisms. We demonstrate on several data sets that Ratatosk decreases the error rate 6-fold and reduces the number of ambiguously mapped bases in the reads. SNPs calls on Ratatosk corrected reads are nearly 99 % accurate and indel calls accuracy is up to 40 % higher compared to the uncorrected reads. Furthermore, variants calls obtained from 4 corrected trios are highly concordant. Finally, we show that Ratatosk corrected data enable highly contiguous assembly with fewer errors compared to other assemblies made from accurate long reads. Future work includes trio-based correction, additional correction passes and an enhanced path coloring of the graph to enable a better correction in repetitive regions.

4 Methods

Section 4.1 details the concepts and data structures that will be used throughout this paper. Section 4.2 describes how the main index is built and preprocessed for correction. Sections 4.3 and 4.4 overview the methods used during the first and second correction passes, respectively.

4.1 Definitions

A string s is a sequence of symbols drawn from an alphabet $\mathcal{A}$. The length of s is denoted by $|s|$. A substring of s is a string in s with a start position i, a length l and is denoted by $s(i, l)$. Let $\mathcal{A}$ be the DNA alphabet $\mathcal{A} = \{A, C, G, T\}$ for which (A, T) and (C, G) are complementing pairs. The reverse-complemented string $\overline{s}$ is the reverse sequence of complemented symbols in s. The canonical string $\hat{s}$ is the lexicographically smallest of s and its reverse-complement $\overline{s}$. A de Bruijn graph (dBG) is a bi-directed graph $G = (V, E)$ in which each vertex $v \in V$ represents a k-mer and its reverse-complement. Only the canonical k-mer of each vertex is stored in G. A directed edge $e \in E$ from vertex v to vertex v' representing k-mers x and x', respectively, exists if and only if $x(2, k-1) = x'(1, k-1)$. Each edge e is labeled with the orientation of the k-mers x and x' they connect: $\{x, x'\}$, $\{x, \overline{x'}\}$, $\{\overline{x}, x'\}$ or $\{\overline{x}, \overline{x'}\}$. Each k-mer x has $|\mathcal{A}|$ possible successors $x(2, k-1) \odot a$ and $|\mathcal{A}|$ possible predecessors $a \odot x(1, k-1)$ in G with $a \in \mathcal{A}$ and $\odot$ as the concatenation operator. The number of k-mers in G is denoted $|G|$. A path in the graph is a sequence of connected vertices $P = (v_1, ..., v_m)$. Path P is said *non-branching* if it is composed of vertices having an in- and out-degree of one with exception of the head vertex v_1 which can have more than one incoming edge and the tail vertex v_m which can have more than one outgoing edge. A non-branching path is maximal if it cannot be extended in the graph without branching. A compacted de Bruijn graph (cdBG) merges all maximal non-branching paths P from the dBG into

single vertices, called *unitigs*, representing substrings of length $|P| + k - 1$. A simplified dBG and its compacted representation are illustrated in Figures 2a and 2b. A colored de Bruijn graph is a graph $G = (V, E, C)$ in which (V, E) is a dBG and C is a set of colors such that each vertex $v \in V$ maps to a subset of C. We extend the definition of a cdBG to a compacted and colored de Bruijn Graph (ccdBG) where (V, E) is a cDBG, so the vertices represent unitigs, and each k-mer of a unitig maps to a subset of C.

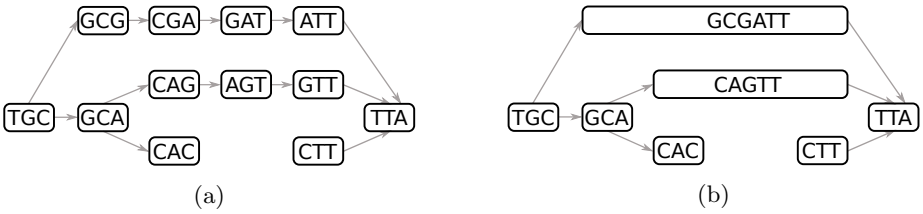


Figure 2: A de Bruijn graph in (a) and its compacted counterpart in (b) using 3-mers. For simplicity, the de Bruijn graph is directed and reverse-complements are not considered.

4.2 Graph construction and preprocessing

Ratatosk takes as input a set $\mathcal{S}$ of paired SRS reads and a set $\mathcal{L}$ of LRS reads. A cdBG is built from $\mathcal{S}$ to correct the reads in $\mathcal{L}$ using two correction passes as shown in Figure 3.

4.2.1 Graph construction

Using different k-mer lengths in the graph built from $\mathcal{S}$ has been shown to improve the correction of $\mathcal{L}$ [Miclotte et al., 2016, Morisse et al., 2018, Wang et al., 2018]: A short k-mer is ideal for finding matches between LRS reads and the graph while unitigs built with long k-mers have a better contiguity. In order to combine the advantages of short and long k-mers, Ratatosk uses two k-mer lengths k_1 and k_2 with $k_2 \geq 2k_1$.

First, a cdBG G_2 is built with the long k_2-mers of $\mathcal{S}$ using the Bifrost graph engine [Holley and Melsted, 2019]. By default, all k_2-mers occurring exactly once in $\mathcal{S}$ are assumed to contain a sequencing error and are discarded from the graph construction. Subsequently, a cdBG G_1 is built from the short k_1-mers of the unitigs in G_2. Graph G_1 is used for the first correction pass while graph G_2 is later used in the second correction pass (Figure 3, Section 4.4).

4.2.2 Graph coloring

Graph G_1 is turned into a ccdBG by coloring its unitigs with the read pairs from $\mathcal{S}$ with which they share at least one k_1-mer, as shown in Figure 4. Given $\frac{|\mathcal{S}|}{2}$ SRS read pairs in input, each pair is identified by a color ranging from 1 to $\frac{|\mathcal{S}|}{2}$. Ratatosk uses hashing to ensure that two reads from the same pair get the same color. Each read is initially given an identifier which is the hash of the read name and each identifier is associated with a color. To simplify the coloring and curb its running time, pair identifiers are not guaranteed to be

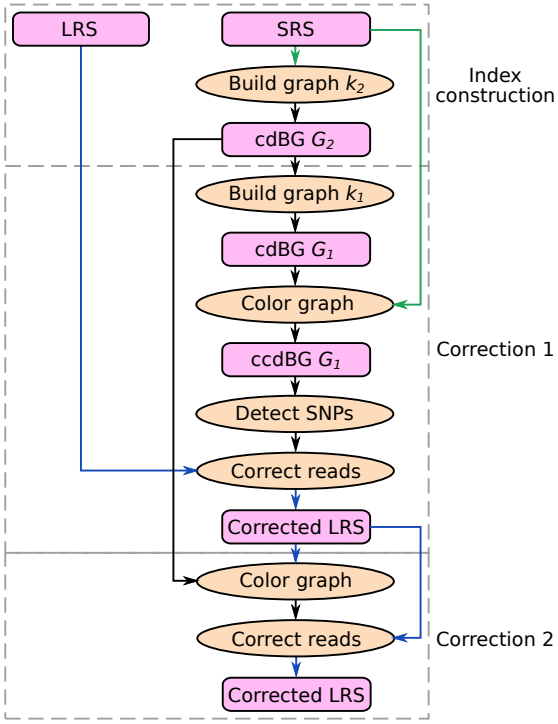


Figure 3: Ratatosk performs two passes of correction, each using a different k-mer size for the graph construction and a different type of reads for the graph coloring. LRS reads are shown in blue and SRS reads in green.

unique such that multiple read pairs might hold the same identifier, hence the same color, because their read names hash to the same value. These collisions are random and are expected to have little impact on the correction.

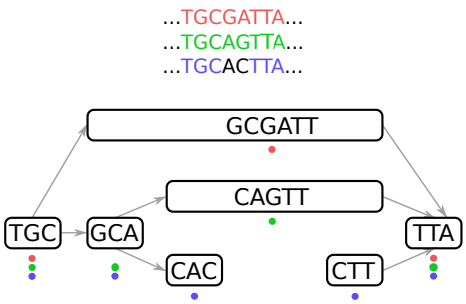


Figure 4: Graph coloring with three colors. Each color represents a read sharing at least one k-mer with a unitig.

Coloring unitigs with read pairs is similar to *partitions* in the guided de Bruijn graph [Holley et al., 2017] and *links* in the Linked de Bruijn graph [Turner et al., 2018]. Ratatosk enables a memory efficient graph coloring by using two techniques. First, it discards *similar* read pairs from the coloring as they rep-

resent redundant information to index, especially in the case of high coverage SRS data. Second, Ratatosk compacts the set of colors assigned to each unitig by using consecutive color values rather than random values. Consecutive color values can be compacted in memory using Run Length Encoding, delta encoding and compacted bitmaps [Chambi et al., 2016].

The graph coloring objective is for each read pair $P_c \in \mathcal{S}$ to color a set of unitigs $\mathcal{U}_c \in G_1$ with color c. Each unitig $u \in \mathcal{U}_c$ shares at least one k_1-mer with P_c as shown in Figure 4. We devised a probabilistic algorithm which colors the graph while discarding similar read pairs and assigning consecutive color values to unitigs in a greedy fashion. The algorithm performs two steps: A partial coloring of the graph is initially made for read pair filtering and color compaction purposes before completing a full graph coloring. In the partial coloring, each read pair P_c will only color the longest unitig $u \in \mathcal{U}_c$. This unitig acts as a centroid in the graph and it is expected that read pairs similar to P_c will color the same unitig. Hence, color c and a hash $h(\mathcal{U}_c)$ are assigned to unitig u unless there exist another pair similar to P_c which has already been assigned to u. Two read pairs P_c and P'_c are similar if they have the same unitig set hash, i.e, $c \neq c'$ but $h(\mathcal{U}_c) = h(\mathcal{U}_{c'})$. In which case, P_c is discarded and will not be used for the full graph coloring. Next, read pairs are given new colors by assigning, when possible, consecutive color values to pairs clustering to the same unitigs. A final graph coloring is then performed using read pairs which are not discarded and their new color values. The mean k_1-mer coverage of unitigs is also computed during the final graph coloring.

After coloring the graph, all unitigs with a mean k_1-mer coverage lower than a pre-defined threshold T_{min} (see Appendix, Section A) are removed from the graph. In order to limit even further the memory usage of Ratatosk, unitigs having a mean k_1-mer coverage greater than T_{max} have their colors discarded. These unitigs are usually k_1-mers with a low sequence entropy such as poly-$\{A, C, G, T\}$ or k_1-mers occurring within STRs. They are typically located within highly branching subgraphs and their colors provide little guidance information while severely impacting the running time.

4.2.3 Candidate SNP annotation

While most *de novo* detection methods for SNPs, indels and SVs are based on the analysis of graph *bubbles* [Onodera et al., 2013, Peterlongo et al., 2017, Paten et al., 2018, Garrison et al., 2018], Ratatosk uses instead a simple but fast string matching method to annotate vertices in the graph containing one or more candidate SNPs. For each k_1-mer x in unitigs, the graph is queried for all k_1-mers having a Hamming distance of 1 with x. Let $x = u(p, k_1)$ and $x' = u'(p', k_1)$ be k_1-mers from unitig u and u', respectively, that differ by exactly one substitution at position $i < k_1$. Unitig u and u' are then annotated at position $p + i$ and $p' + i$, respectively, with a IUPAC symbol representing the substitution. For example, symbol R would be assigned to position 3 in unitigs GCGATT and GCA of Figure 4 to represent an A/G substitution.

4.3 First correction pass

The following section describes how LRS reads are anchored to the ccdBG and the methods used to correct non-anchored regions of the LRS reads.

4.3.1 Read Anchoring

We define *solid* and *weak* k-mers similarly as defined in LoRDEC and introduce the definition of *near solid* k-mers:

- solid k-mer: exact length k substring match between a long read and a unitig from the graph.
- near solid k-mer: inexact length k substring match between a long read and a unitig from the graph with one base substitution or indel.
- weak k-mer: length k substring of a long read which is neither a solid k-mer nor a near solid k-mer.

We define two types of regions in a long read:

- solid region: a region of a long read composed only of solid k-mers.
- non-solid region: a region of a long read composed of weak or near solid k-mers.

A solid or near solid k-mer is also called a *match*. A match between long read r at position p_r and unitig u at position p_u is denoted $m = \langle p_r, r, p_u, u \rangle$. A match m is *unique* if it is the only match at position p_r in r. A k-mer has at most one solid match in G_1 but can have multiple near solid matches in G_1. Note that solid and non-solid regions can overlap by $k-1$ bases. All non-solid regions are surrounded by two solid regions with the exception of non-solid regions at the start and end of LRS reads.

4.3.2 Delimiting non-solid regions

Each read $r \in \mathcal{L}$ is corrected independently, allowing multiple threads to correct LRS reads in parallel. The graph is queried for each k_1-mer of r, resulting in a list of solid matches M_s and a list of near solid matches M_n, both sorted by ascending match position p_r in r. Only unique near solid matches (UNSM) are kept in M_n to prevent anchoring r on a SNP or indel from an incorrect allele. Furthermore, a k_1-mer which is both a solid match and a near solid match is considered solid and its near solid matches are discarded from M_n.

Non-solid regions of r are detected by finding all pairs of successive solid matches $m_s, m_t \in M_s$ for which $p_r^s \neq p_r^t - 1$ with the exception of non-solid regions at the extremities of r. The first match m_s of the pair is referred to as the *source* match and the second match m_t of the pair is referred to as the *target* match. The length of the non-solid region to correct is then $l = p_r^t - p_r^s + k_1$. It includes $r(p_r^s, k_1)$ which is the last solid k_1-mer from the source solid region and $r(p_r^t, k_1)$ which is the first solid k_1-mer from the target solid region as illustrated in Figure 5. If a read starts with a non-solid region, that region has no source match and hence starts on the first position of the read. Similarly, if a read ends with a non-solid region, that region has no target match and hence ends on the last position of the read.

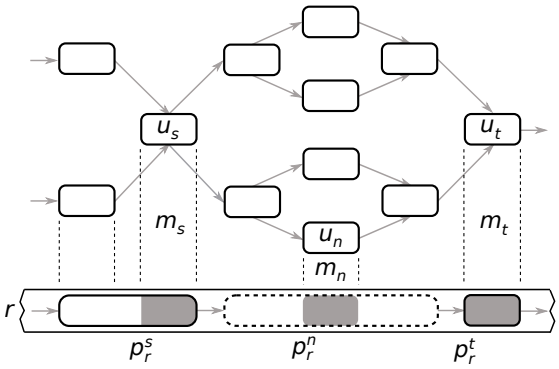


Figure 5: Example of a long read r anchored on a ccdBG. A section of r is shown at the bottom with two solid regions (non-dashed boxes at the extremities) surrounding a non-solid region (dashed line box at the center). The grey areas of the solid regions show the source and target matches between the long read and the graph. The grey area of the non-solid region shows a near solid match. For simplicity, colors are not shown.

4.3.3 Traversing the graph

In order to correct a non-solid region, Ratatosk attempts to extract one path in the graph connecting unitig u_s of the source match to unitig u_t of the target match. Since the length l of the non-solid region to correct is known, we assume that the corrected path between u_s and u_t has minimum sequence length $l_{min} = \frac{l}{1+F}$ bases and maximum sequence length $l_{max} = l \cdot (1 + F)$ bases where F is an upper-bound of the error rate in the long read (see Appendix, Section A). Ratatosk uses two greedy techniques to guide the traversal in the graph and prune the search space, as shown in Figure 6.

First, rather than exploring all paths between unitigs u_s and u_t, Ratatosk only explores paths traversing UNSMs in the non-solid region to correct. These matches provide an anchoring in the non-solid regions as they are near exact k_1-mer matches between the graph and the read to correct. Hence, paths between u_s and u_t which do not traverse the UNSMs are pruned because they are not good candidates for the correction. Let m_n be the near solid match from M_n with the smallest position p_r^n such that $p_r^s + k_1 \leq p_r^n \leq p_r^t - k_1$. Ratatosk first attempts to extract one path connecting unitig u_s to unitig $u_n \in m_n$ with a BFS traversal that only explores paths with maximum sequence length $(p_r^n - p_r^s + k_1) \cdot (1 + F)$ bases. The process of extending the last unitig of a path to the next UNSM in M_n is repeated until there are no more UNSMs to consider in M_n or no path extension is possible. Finally, the graph traversal attempts to extend the path to the target unitig u_t. Note that in the absence of UNSM in the non-solid region to correct, all paths connecting u_s and u_t with minimum sequence length l_{min} and maximum sequence length l_{max} are traversed.

Second, even using UNSMs to prune the search space during traversal, the subgraph between two unitigs u_n and $u_n{}'$ from UNSMs can be very large. This is

particularly true for LRS reads with a high error rate, resulting in long non-solid regions with few or no UNSMs. In order to prune the search space between u_n and $u_n{}'$, a greedy graph traversal is used to extract one path connecting the two unitigs. Unitig u_n is first extended by visiting all paths of length P_{max} vertices with a BFS traversal. Each traversed path is given a probability s_P of being the correct path to extend and only the path with the greatest probability is extended. The path chosen for extension maximizes its sequence similarity with the non-solid region to correct. Furthermore, as colors highlight paths in the graph representing SRS reads, the path chosen for extension also maximizes its color similarity with the surrounding solid regions. Hence, before correcting a non-solid region, Ratatosk first computes the union C of all colors sets C_u from the solid matches and UNSMs within an interval corresponding to the non-solid region start and end positions extended of B bases on each side, i.e ,

$$C = \bigcup_{u \in m} C_u, \forall m \in M_s, M_n \text{ with } p_r^s - B \leq p_r \leq p_r^t + B \tag{1}$$

During the BFS traversal, a path probability s_P is computed for each traversed path based on the number of colors the path shares with C and the sequence similarity of the path to the region to correct. Specifically, given a path P composed of P_{max} unitigs and its color set $C_p = \bigcup_{u \in P} C_u$, the color matching probability of P is $s_c = \frac{|C_p \cap C|}{|C|}$ and the sequence matching probability s_q is derived from the normalized edit distance of P to the non-solid region to correct using an infix alignment computed by the edlib tool [Šošić and Šikić, 2017]. Both probabilities are then conflated:

$$s_P = \frac{s_c \cdot s_q}{s_c \cdot s_q + (1 - s_c) \cdot (1 - s_q)} \tag{2}$$

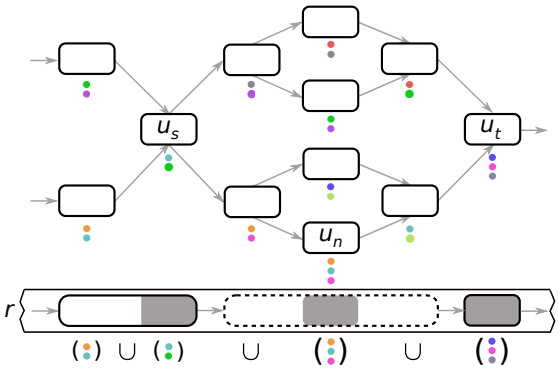
The path with the greatest probability s_P is extended by starting a new graph traversal from its last unitig. The extension continues until unitig $u_n{}'$ is reached or no path can be extracted as a result of a tip in the graph or extending over $(p_r^{n'} - p_r^n + k) \cdot (1 + F)$ bases.

To enable a faster traversal, a local minimum number of colors T_C is computed from the surrounding solid regions and the unitigs of UNSMs. Each traversed unitig u of a path P must be colored by at least T_C colors of C such that:

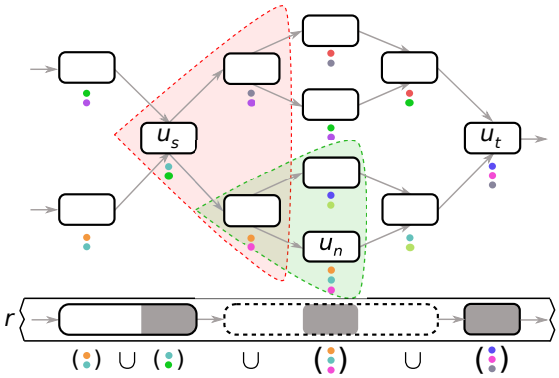
$$T_C = D \cdot \min_{u \in m} |C_u|, \forall m \in M_s, M_n \text{ with } p_r^s - B \leq p_r \leq p_r^t + B \tag{3}$$

and D being a fixed lower bound factor (see Appendix, Section A). If the color set of a traversed unitig has less than T_C colors, its path is not explored any further nor it is considered for extension.

A path extension connecting unitig u_u to unitig u_v might end prematurely for multiple reasons: all possible extensions end on a tip of the graph because of incomplete SRS data or insufficient color coverage in the traversed subgraph. In such a case, the extended path is completed with a gap corresponding to the non-solid subsequence to correct and the path extension resumes from unitig u_v.



(a)



(b)

Figure 6: In (a), the union of colors is computed within the solid regions around the non-solid region to correct and the USNMs. This union will partially guide the graph traversal, along with the sequence similarity of the paths to the non-solid region. In (b), a first subgraph (highlighted in red) of all paths starting at unitig u_s with $P_{max} = 2$ unitigs is explored for correction. The lower path is extended using the same method (shown in green) and a path connecting to u_n is found.

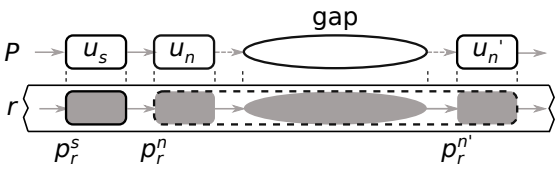


Figure 7: Example of a gap in a path. Path P is first extended until unitig u_n, then a gap corresponding to a subsequence from the uncorrected read is inserted in P and the extension of P resumes from unitig $u_n{}'$.

An example of path extension with a gap is illustrated in Figure 7.

Finally, non-solid regions located on long read extremities have only one surrounding solid region. The non-solid region at the start of a long read is

corrected using a backward graph traversal from u_t and the one at the end of a long read is corrected with a forward graph traversal from u_s. Because each of these graph traversals has no target match, any path with length l base such that $l_{min} \leq l \leq l_{max}$ is returned as a candidate for correction.

4.3.4 Forward and backward corrections

A candidate path for correction is *incomplete* if it contains a gap or if it does not connect to the unitig of a target match. If no path or only an incomplete path has been extracted, Ratatosk corrects the non-solid region backward, i.e., from the target match to the source match. Indeed, the forward graph traversal might have stopped prematurely for multiple reasons, one of which being that the color guidance led incorrectly to a tip in the graph. However, traversing the graph backward might lead to a different path. If both forward and backward paths are incomplete, Ratatosk merges both paths by aligning their sequences to the non-solid region using the Needleman–Wunsch algorithm (global alignment). The merged sequence is created by traversing the alignment of both forward and backward corrections at the same time and selecting subsequences in each corrections. In the case of candidate paths starting or ending a long read, all candidate paths are aligned to the non-solid region using a local alignment that does not penalize gaps at the end. The candidate path with the smallest edit distance is chosen for the correction.

4.3.5 Candidate SNP correction

Heuristics used to traverse the graph as presented in Section 4.3.3 might incorrectly extend a path and lead to the erroneous correction of a non-solid region using SNPs from incorrect alleles. Once a path has been selected to correct a non-solid region, all the positions in this path matching candidate SNPs and their IUPAC symbols are known from the unitigs. Let s be the non-solid region and s' its corrected counterpart. Sequence s' is aligned to s and a CIGAR string is generated from the alignment. Ratatosk iterates over matching positions of the CIGAR string (symbol M) denoted $m = \langle s, p, s', p' \rangle$. Note that m indicates that base $s(p, 1)$ is either a match or a mismatch with base $s'(p', 1)$ but is not part of an insertion or deletion in the alignment. Let M_{snp} be the set of all matches $m = \langle s, p, s', p' \rangle$ for which $s'(p', 1)$ has an assigned IUPAC symbol in the graph indicating a candidate SNP. For each match $m = \langle s, p, s', p' \rangle \in M_{snp}$, base $b = s(p, 1)$ is compared to the IUPAC symbol associated to $b' = s'(p', 1)$. If b is one of the possible bases represented by the IUPAC symbol, then b' is corrected with b. This method enables a conservative correction of SNPs in the corrected non-solid regions by using only bases from the uncorrected non-solid regions which are compatible with the candidate SNPs from the graph. However, this method only corrects SNPs in the matching or mismatching regions of the alignment and discards candidate SNPs located within insertions of s'. To overcome this issue, a match $m \in M_{snp}$ is said *strongly compatible* if $s'(p', 1) = s(p, 1)$ prior to SNP correction. A strongly compatible SNP indicates that Ratatosk is confident in the subpath that was selected to correct the region around that candidate SNP. As the strongly compatible SNP at position p' is from unitig $u' \in m$, all bases which are candidate SNPs in u' are used to correct SNPs in the inserted positions of the alignment (symbol I in the CIGAR string)

around position p'.

4.4 Second correction pass

In the first correction pass, Ratatosk corrected each LRS read independently from the other reads in $\mathcal{L}$. In a second correction pass, Ratatosk takes advantage of the set of corrected LRS reads as a whole. Indeed, reads corrected during the first pass might be sufficiently error-free to correct the remaining non-solid regions. Furthermore, LRS reads are at least an order of magnitude longer than SRS reads and do not need to be paired, hence offering more information to which paths to traverse in the graph. In the following, we describe the second correction pass with a highlight on the differences with the first correction pass.

Let $\mathcal{L}'$ be the set of corrected LRS reads obtained from the first correction pass. First, graph G_2 built from the k_2-mers of $\mathcal{S}$ (Section 4.2.1) is loaded in memory. Compared to G_1, unitigs of G_2 have a better contiguity and some of the highly branching subgraphs of G_1 corresponding to repetitive regions are untangled in G_2. Graph coloring and candidate SNP annotation using $\mathcal{L}'$ are performed as described in Sections 4.2.2 and 4.2.3, respectively. Because the reads in $\mathcal{L}'$ are long and still erroneous in the uncorrected regions, they are not expected to be similar and Ratatosk does not perform similar reads removal.

Reads of $\mathcal{L}'$ are then anchored on the graph and non-solid regions are corrected as described in Section 4.3. Parameter B in Equation 1 corresponds to the size of a buffer around a non-solid region where the union of unitig colors from solid and UNSMs is computed. In the first correction pass, solid regions are expected to be short and sparse because of the high error rate of LRS reads. Hence, B was large enough to span two SRS reads from the same pair and the gap that intersperse them in order to capture as many colors as possible. Corrected LRS reads have no gap and are much longer than SRS reads so it is expected that solid regions are much more abundant and contiguous than during the first correction pass. Distance B is therefore much smaller for the second pass (Appendix, Section A) which saves computation time. Furthermore, solid regions are required to be at least $B > k_2$ bases long in the second pass to increase the contiguity of solid regions and provide a better anchoring on the graph.

During path selection described in Section 4.3.3, BFS traversals explored all paths of P_{max} unitigs and a path probability was assigned to each one of them before selecting one path for extension. Traversing a fixed number of unitigs avoids a combinatorial growth of the number of explored paths, especially in complex subgraphs with short cycles that are characteristic of STRs. However, as unitigs can have any length $\geq k_1$, it has the disadvantage that the path probability might be computed for paths of P_{max} unitigs with different sequence lengths. Instead, the graph traversal in the second correction pass explores paths with a minimum sequence length of B bases rather than a minimum number of unitigs.

Once a path P has at least B bases in its sequence, its color matching probability s_c and sequence matching probability s_q are computed and conflated into a path probability s_P. The construction of color set C used in the color

matching probability s_c is shown in Equation 4 and only uses the intersection of colors from each side of the non-solid region, i.e , C^s and C^t, rather than the union (Equation 1) in order to remove erroneous colors which do not belong to this region:

$$\begin{aligned} C &= C^s \bigcup C^t \\ C^s &= \bigcap_{u \in m} C_u,\ \forall m \in M_s \text{ with } p^s_{r'} - B \leq p_{r'} \leq p^s_{r'} \\ C^t &= \bigcap_{u \in m} C_u,\ \forall m \in M_s \text{ with } p^t_{r'} \leq p_{r'} \leq p^t_{r'} + B \end{aligned} \tag{4}$$

4.5 Reference-guided correction

While Ratatosk is a reference-free method, we propose an optional reference-guided preprocessing of the reads which is beneficial in several ways. This pipeline first maps the input SRS and LRS reads to a reference genome and then clusters the reads into *bins* corresponding to 5 Mbp long regions of the reference. Each bin of SRS and LRS reads is subsequently corrected independently. The benefit is three-fold:

- Graphs G_1 and G_2 built from an SRS bin are much smaller and contiguous than for the entire SRS data set, hence reducing the probability of selecting an incorrect path during graph traversal.
- Computation time is reduced as the search space in each bin is much smaller than for the entire SRS data set.
- Each bin is corrected independently so the workload can be distributed in parallel over many nodes of an HPC.

However, a reference-guided preprocessing also introduces some challenges. First, it is common that reference genomes contain gaps. For example, the human genome reference GRCh38.p13 has about 161 Mbp of N bases. Second, SRS reads overlapping large insertion events are expected to be unmapped. Finally, SRS reads with poor mapping qualities map ambiguously to the reference and might be incorrectly binned.

To overcome these issues, Ratatosk detects reads from the full SRS data set $\mathcal{S}$ which are likely missing in each bin. Let $\mathcal{S}_b$ and $\mathcal{L}_b$ be the subset of SRS and LRS reads of a bin b, respectively. To begin with, cdBGs $G_b^{\mathcal{S}}$ and $G_b^{\mathcal{L}}$ are built from the k_1-mers occurring twice or more in $\mathcal{S}_b$ and $\mathcal{L}_b$, respectively. Once $G_b^{\mathcal{L}}$ is built, its unitigs are annotated with their mean k_1-mer coverage. At first, $G_b^{\mathcal{L}}$ contains many more k_1-mers than $G_b^{\mathcal{S}}$ because many erroneous k_1-mers from $\mathcal{L}_b$ occur twice or more in the bin. To prune these erroneous k_1-mers from $G_b^{\mathcal{L}}$, unitigs having low coverages are removed iteratively until $|G_b^{\mathcal{L}}| \approx |G_b^{\mathcal{S}}|$. Subsequently, all reads $r \in \mathcal{S}$ are queried: If r contains many k_1-mers occurring in $G_b^{\mathcal{L}}$ but not in $G_b^{\mathcal{S}}$, r is suspected to be missing from the bin and is added to $\mathcal{S}_b$.

We outline the binning and correction pipeline proposed, as illustrated in Figure 8, in the following. First, all reads from $\mathcal{S}$ and $\mathcal{L}$ are binned into regions

of 5 Mbp according to their mapping to the reference genome. Low mapping quality (< 30) and unmapped LRS reads are set aside in a bin for ambiguous long reads. Once all reads have been binned, a local correction is performed in parallel for all non-ambiguous bins and the output corrected LRS reads are merged. Note that each bin correction has access to $\mathcal{S}$ (top red arrows in Figure 8) to retrieve the missing SRS reads from the bin. Finally, the bin of ambiguous LRS reads is corrected globally using $\mathcal{S}$. This correction is assisted by the previously corrected non-ambiguous LRS reads to enhance graph coloring during the second round of correction (bottom red arrow in Figure 8).

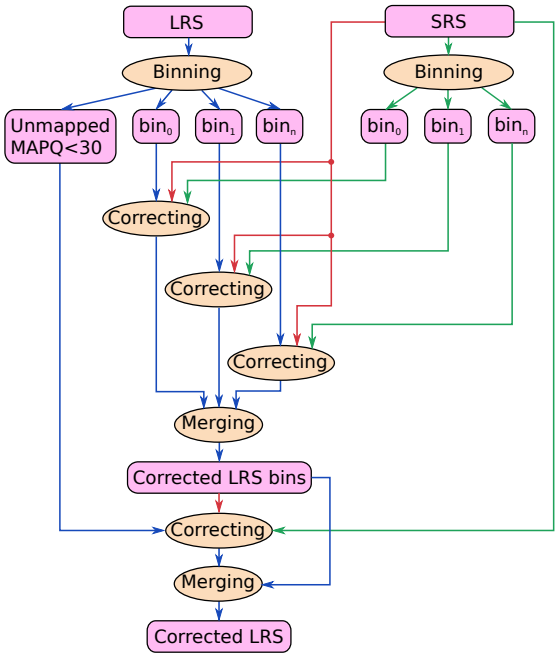


Figure 8: Reference-guided preprocessing of the input SRS reads (green) and LRS reads (blue). Reads are first binned and each bin is corrected independently. Unmapped or low mapping quality LRS reads are corrected using all SRS reads and all corrected LRS bins. Red arrows indicate input read sets which assist with the correction but are not corrected themselves.

Acknowledgements

The authors would like to thank our colleagues from deCODE genetics and Amgen Inc. as well as Peter Loof Møller from Aarhus University for their helpful feedback during the development of Ratatosk. We would also like to thank all research participants who provided a biological sample to deCODE genetics and to Genome In A Bottle.

Author's contributions

GH implemented the Ratatosk software. GH and BVH designed the Ratatosk algorithm with input from DB, HI, SK and HPE. GH and BVH designed the experiments. GH analyzed the data sets. GH wrote the initial version of the manuscript. All authors contributed to subsequent versions. All authors reviewed and approved the final version of the manuscript.

Competing interests

All authors are employees of deCODE Genetics/Amgen Inc.

References

P. A. Audano, A. Sulovari, T. A. Graves-Lindsay, S. Cantsilieris, M. Sorensen, A. E. Welch, M. L. Dougherty, B. J. Nelson, A. Shah, S. K. Dutcher, W. C. Warren, V. Magrini, S. D. McGrath, Y. I. Li, R. K. Wilson, and E. E. Eichler. Characterizing the Major Structural Variant Alleles of the Human Genome. *Cell*, 176(3):663–675.e19, 2019. doi: https://doi.org/10.1016/j.cell.2018.12.019.

D. Beyter, H. Ingimundardottir, H. P. Eggertsson, E. Bjornsson, S. Kristmundsdottir, S. Mehringer, H. Jonsson, M. T. Hardarson, D. N. Magnusdottir, R. P. Kristjansson, S. A. Gudjonsson, S. T. Sverrisson, G. Holley, G. Eyjolfsson, I. Olafsson, O. Sigurdardottir, G. Masson, U. Thorsteinsdottir, D. F. Gudbjartsson, P. Sulem, O. T. Magnusson, B. V. Halldorsson, and K. Stefansson. Long read sequencing of 1,817 icelanders provides insight into the role of structural variants in human disease. *bioRxiv*, 2019. doi: 10.1101/848366.

A. V. Bzikadze and P. A. Pevzner. centroFlye: Assembling Centromeres with Long Error-Prone Reads. *bioRxiv*, 2019. doi: 10.1101/772103.

S. Chambi, D. Lemire, O. Kaser, and R. Godin. Better bitmap performance with Roaring bitmaps. *Software: Practice and Experience*, 46(5):709–719, 2016.

Y.-C. Chen, T. Liu, C.-H. Yu, T.-Y. Chiang, and C.-C. Hwang. Effects of GC bias in next-generation-sequencing data on de novo genome assembly. *PLOS One*, 8(4), 2013.

P. Edge and V. Bansal. Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing. *Nat. Commun.*, 10(4660), 2019.

S. Fu, A. Wang, and K. F. Au. A comparative evaluation of hybrid error correction methods for error-prone long reads. *Genome Biol.*, 20(26), 2019.

E. Garrison, J. Sirén, A. M. Novak, G. Hickey, J. M. Eizenga, E. T. Dawson, W. Jones, S. Garg, C. Markello, M. F. Lin, B. Paten, and R. Durbin. Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nat. Biotechnol.*, 36:875–879, 2018.

A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler. Quast: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

E. Haghshenas, F. Hach, S. C. Sahinalp, and C. Chauve. CoLoRMap: Correcting Long Reads by Mapping short reads. *Bioinformatics*, 32(7):i545–i551, 2015.

G. Holley and P. Melsted. Bifrost - Highly parallel construction and indexing of colored and compacted de Bruijn graphs. *bioRxiv*, 2019. doi: 10.1101/695338.

G. Holley, R. Wittler, J. Stoye, and F. Hach. Dynamic Alignment-Free and Reference-Free Read Compression. In *Proc. of the 21st International Conference on Research in Computational Molecular Biology (RECOMB'17)*, volume 10229 of *Lecture Notes in Computer Science*, pages 50–65, 2017.

R. M. Idury and M. S. Waterman. A new algorithm for DNA sequence assembly. *J. Comput. Biol.*, 2(2), 1995.

H. Jonsson, P. Sulem, B. Kehr, S. Kristmundsdottir, F. Zink, E. Hjartarson, M. T. Hardarson, K. E. Hjorleifsson, H. P. Eggertsson, S. A. Gudjonsson, L. D. Ward, G. A. Arnadottir, E. A. Helgason, H. Helgason, A. Gylfason, A. Jonasdottir, A. Jonasdottir, T. Rafnar, S. Besenbacher, M. L. Frigge, S. N. Stacey, O. T. Magnusson, U. Thorsteinsdottir, G. Masson, A. Kong, B. V. Halldorsson, A. Helgason, D. F. Gudbjartsson, and K. Stefansson. Whole genome characterization of sequence diversity of 15,220 Icelanders. *Sci. Data*, 4:170115, 2017.

M. Kolmogorov, J. Yuan, Y. Lin, and P. A. Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.*, 37:540–546, 2019.

I. Kozarewa, Z. Ning, M. A. Quail, M. J. Sanders, M. Berriman, and D. J. Turner. Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+ C)-biased genomes. *Nat. Methods*, 6:291–295, 2009.

S. Kristmundsdottir, H. P. Eggertsson, G. A. Arnadottir, and B. V. Halldorsson. popSTR2 enables clinical and population-scale genotyping of microsatellites. *Bioinformatics*, 36(7):2269–2271, 2020.

P. Krusche, L. Trigg, P. C. Boutros, C. E. Mason, M. Francisco, B. L. Moore, M. Gonzalez-Porta, M. A. Eberle, Z. Tezak, S. Lababidi, R. Truty, G. Asimenos, B. Funke, M. Fleharty, B. A. Chapman, M. Salit, J. M. . t. G. A. f. G. Zook, and H. B. Team. Best practices for benchmarking germline small-variant calls in human genomes. *Nat. Biotechnol.*, 37:555–560, 2019.

H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. Subgroup. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

G. A. Logsdon, M. R. Vollger, and E. E. Eichler. Long-read human genome sequencing and its applications. *Nat. Rev. Genet.*, 2020.

R. Luo, C.-L. Wong, Y.-S. Wong, C.-I. Tang, C.-M. Liu, C.-M. Leung, and T.-W. Lam. Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nat. Mach. Intell.*, 2:220–227, 2020.

C. Marchet, P. Morisse, L. Lecompte, A. Lefebvre, T. Lecroq, P. Peterlongo, and A. Limasset. ELECTOR: Evaluator for long reads correction methods. *NAR Genomics and Bioinformatics*, 2(1):lqz015, 2020.

P. Marijon, R. Chikhi, and J.-S. Varré. yacrd and fpa: upstream tools for long-read genome assembly. *Bioinformatics*, 2020. doi: 10.1093/bioinformatics/btaa262. btaa262.

G. Miclotte, M. Heydari, P. Demeester, S. Rombauts, Y. Van de Peer, P. Audenaert, and J. Fostier. Jabba: hybrid error correction for long sequencing reads. *Algorithms Mol. Biol.*, 11(10), 2016.

K. H. Miga. Centromeric satellite DNAs: hidden sequence variation in the human population. *Genes*, 10(5):352, 2019.

K. H. Miga, S. Koren, A. Rhie, M. R. Vollger, A. Gershman, A. Bzikadze, S. Brooks, E. Howe, D. Porubsky, G. A. Logsdon, V. A. Schneider, T. Potapova, J. Wood, W. Chow, J. Armstrong, J. Fredrickson, E. Pak, K. Tigyi, M. Kremitzki, C. Markovic, V. Maduro, A. Dutra, G. G. Bouffard, A. M. Chang, N. F. Hansen, F. Thibaud-Nissen, A. D. Schmitt, J.-M. Belton, S. Selvaraj, M. Y. Dennis, D. C. Soto, R. Sahasrabudhe, G. Kaya, J. Quick, N. J. Loman, N. Holmes, M. Loose, U. Surti, R. a. Risques, T. A. Graves Lindsay, R. Fulton, I. Hall, B. Paten, K. Howe, W. Timp, A. Young, J. C. Mullikin, P. A. Pevzner, J. L. Gerton, B. A. Sullivan, E. E. Eichler, and A. M. Phillippy. Telomere-to-telomere assembly of a complete human X chromosome. *Nature*, 2020. doi: 10.1038/s41586-020-2547-7.

S. Mitsuhashi, M. C. Frith, T. Mizuguchi, S. Miyatake, T. Toyota, H. Adachi, Y. Oma, Y. Kino, H. Mitsuhashi, and N. Matsumoto. Tandem-genotypes: robust detection of tandem repeat expansions from long dna reads. *Genome Biol.*, 20(58), 2019.

P. Morisse, T. Lecroq, and A. Lefebvre. Hybrid correction of highly noisy long reads using a variable-order de bruijn graph. *Bioinformatics*, 34(24):4213–4222, 2018.

P. Morisse, T. Lecroq, and A. Lefebvre. Long-read error correction: a survey and qualitative comparison. *bioRxiv*, 2020. doi: 10.1101/2020.03.06.977975.

S. Nurk, B. P. Walenz, A. Rhie, M. R. Vollger, G. A. Logsdon, R. Grothe, K. H. Miga, E. E. Eichler, A. M. Phillippy, and S. Koren. HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *bioRxiv*, 2020. doi: 10.1101/2020.03.14.992248.

T. Onodera, K. Sadakane, and T. Shibuya. Detecting superbubbles in assembly graphs. In *Proc. of the 13th Workshop on Algorithms in Bioinformatics (WABI'13)*, volume 8126, pages 338–348, 2013.

Oxford Nanopore Technologies. *Medaka*, accessed June 10th 2020, 2019. https://nanoporetech.github.io/medaka/snp.html#snp-and-indel-calling.

B. Paten, J. M. Eizenga, Y. M. Rosen, A. M. Novak, E. Garrison, and G. Hickey. Superbubbles, ultrabubbles, and cacti. *J. Comput. Biol.*, 25(7):649–663, 2018.

P. Peterlongo, C. Riou, E. Drezen, and C. Lemaitre. DiscoSnp++: de novo detection of small variants from raw unassembled read set(s). *bioRxiv*, 2017. doi: 10.1101/209965.

P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. USA*, 98(17):9748–9753, 2001.

R. Poplin, P.-C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, J. Dijamco, N. Nguyen, P. T. Afshar, S. S. Gross, L. Dorfman, C. Y. McLean, and M. A. DePristo. A universal SNP and small-indel variant caller using deep neural networks. *Nat. Biotechnol.*, 36:983–987, 2018.

D. Porubsky, P. Ebert, P. A. Audano, M. R. Vollger, W. T. Harvey, K. M. Munson, M. Sorensen, A. Sulovari, M. Haukness, M. Ghareghani, P. M. Lansdorp, B. Paten, S. E. Devine, A. D. Sanders, C. Lee, M. J. Chaisson, J. O. Korbel, E. E. Eichler, and T. Marschall. A fully phased accurate assembly of an individual human genome. *bioRxiv*, 2019. doi: 10.1101/855049.

F. J. Rang, W. P. Kloosterman, and J. de Ridder. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome Biol.*, 19(1):90, 2018.

A. Rhie, B. P. Walenz, S. Koren, and A. M. Phillippy. Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *bioRxiv*, 2020. doi: 10.1101/2020.03.15.992941.

L. Salmela and E. Rivals. LoRDEC: accurate and efficient long read error correction. *Bioinformatics*, 30(24):3506–3514, 2014.

F. J. Sedlazeck, H. Lee, C. A. Darby, and M. C. Schatz. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nat. Rev. Genet.*, 19:329–346, 2018a.

F. J. Sedlazeck, P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. von Haeseler, and M. C. Schatz. Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods*, 15:461–468, 2018b.

K. Shafin, T. Pesout, R. Lorig-Roach, M. Haukness, H. E. Olsen, C. Bosworth, J. Armstrong, K. Tigyi, N. Maurer, S. Koren, F. J. Sedlazeck, T. Marschall, S. Mayes, V. Costa, J. M. Zook, K. J. Liu, D. Kilburn, M. Sorensen, K. M. Munson, M. R. Vollger, J. Monlong, E. Garrison, E. E. Eichler, S. Salama, D. Haussler, R. E. Green, M. Akeson, A. Phillippy, K. H. Miga, P. Carnevali, and M. Jain. Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nat. Biotechnol.*, 2020.

A. Shumate, A. V. Zimin, R. M. Sherman, D. Puiu, J. M. Wagner, N. D. Olson, M. Pertea, M. L. Salit, J. M. Zook, and S. L. Salzberg. Assembly and annotation of an Ashkenazi human reference genome. *Genome Biol.*, 21, 2020.

M. Šošić and M. Šikić. Edlib: a c/c++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, 33(9):1394–1395, 2017.

I. Turner, K. V. Garimella, Z. Iqbal, and G. McVean. Integrating long-range connectivity information into de Bruijn graphs. *Bioinformatics*, 34(15):2556–2565, 2018.

J. R. Wang, J. Holt, L. McMillan, and C. D. Jones. FMLRC: Hybrid long read error correction using an FM-index. *BMC Bioinform.*, 19:50, 2018.

A. M. Wenger, P. Peluso, W. J. Rowell, P.-C. Chang, R. J. Hall, G. T. Concepcion, J. Ebler, A. Fungtammasan, A. Kolesnikov, N. D. Olson, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat. Biotechnol.*, 37:1155–1162, 2019.

H. Zhang, C. Jain, and S. Aluru. A comprehensive evaluation of long read error correction methods. *bioRxiv*, 2019. doi: 10.1101/519330.

A. V. Zimin, G. Marçais, D. Puiu, M. Roberts, S. L. Salzberg, and J. A. Yorke. The MaSuRCA genome assembler. *Bioinformatics*, 29(21):2669–2677, 2013.

J. M. Zook, D. Catoe, J. McDaniel, L. Vang, N. Spies, A. Sidow, Z. Weng, Y. Liu, C. E. Mason, N. Alexander, E. Henaff, A. B. R. McIntyre, D. Chandramohan, F. Chen, E. Jaeger, A. Moshrefi, K. Pham, W. Stedman, T. Liang, M. Saghbini, Z. Dzakula, A. Hastie, H. Cao, G. Deikus, E. Schadt, R. Sebra, A. Bashir, R. M. Truty, C. C. Chang, N. Gulbahce, K. Zhao, S. Ghosh, F. Hyland, Y. Fu, M. Chaisson, C. Xiao, J. Trow, S. T. Sherry, A. W. Zaranek, M. Ball, J. Bobe, P. Estep, G. M. Church, P. Marks, S. Kyriazopoulou-Panagiotopoulou, G. X. Y. Zheng, M. Schnall-Levin, H. S. Ordonez, P. A. Mudivarti, K. Giorda, Y. Sheng, K. Bjarnesdatter Rypdal, and M. Salit. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Sci. Data*, 3:160025, 2016.

A Default parameters

Graph construction and coloring:

- $k_1 = 31$
- $k_2 = 63$
- $F = 0.25$
- $T_{max}(u) = 512 \cdot (|u| - k + 1)$ for $u \in V$

First correction pass:

- $T_{min} = 2$
- $B = 500$
- $D = 0.1$
- $P_{max} = 4$

Second correction pass:

- $T_{min} = 1$
- $B = 95$

B Error rate

Let r be an LRS read which has been aligned to a reference genome. We define the following:

- $|r|$: Number of bases in r
- D_r: Number of deleted bases in alignment of r to reference
- I_r: Number of inserted bases in alignment of r to reference
- M_r: Number of mismatching bases in alignment of r to reference
- S_r: Number of soft-clipped bases in alignment of r to reference

The deletion, insertion and substitution error rates of a set of LRS reads $\mathcal{L}$ are:

$$\begin{aligned} E_D &= \sum_{r \in \mathcal{L}} \frac{D_r}{|r| - S_r} \\ E_I &= \sum_{r \in \mathcal{L}} \frac{I_r}{|r| - S_r} \\ E_M &= \sum_{r \in \mathcal{L}} \frac{M_r}{|r| - S_r} \end{aligned} \tag{5}$$

And the combined error rate of R is:

$$E = \sum_{r \in \mathcal{L}} \frac{D_r + I_r + M_r}{|r| - S_r} \tag{6}$$

The error rates are computed from primary alignments only.

C Ambiguous bases

Ambiguous bases are bases which soft-clip in the primary alignment but map in non-overlapping supplementary alignments to distant reference positions, such as different chromosomes, compared to the primary alignment reference position. The ratio of ambiguous bases aim to measure the proportion of over-corrected bases in reads. To avoid counting ambiguous bases in supplementary alignments of a read r that might correspond to a large SV event, a supplementary alignment is only used if it does not overlap the primary alignment nor another supplementary alignment of r with a large buffer of 1 Mbp on each side of the alignment. Algorithm 1 details the ambiguous bases ratio computation for a read represented by a primary alignment and a set of supplementary alignments.

Algorithm 1 Compute ratio of ambiguous bases

Input: Read r, Primary alignment P, list of supplementary alignments $\mathcal{S}$

function AMBIGUOUSBASES(r, P, $\mathcal{S}$)
  $\mathcal{S}' \leftarrow \mathsf{sort}(\mathcal{S})$ ▷ Sort by decreasing alignment score
  $\mathcal{C}_a \leftarrow \varnothing$ ▷ Set of ambiguous read positions
  $\mathcal{C}_P \leftarrow$ set of soft-clipped read positions in P
  $B \leftarrow 1,000,000$ ▷ Buffer size in bp
  $T \leftarrow$ empty interval tree
  $p_s^P \leftarrow$ first mapped position of P in reference
  $p_e^P \leftarrow$ last mapped position of P in reference
  $T.\mathsf{addInterval}(p_s^P, p_e^P)$ ▷ Add $[p_s^P, p_e^P]$
  for each $S \in \mathcal{S}'$ **do**
    $p_s^S \leftarrow$ first mapped position of S in reference
    $p_e^S \leftarrow$ last mapped position of S in reference
    $\mathcal{O} \leftarrow T.\mathsf{getOverlappingIntervals}(p_s^S - B, p_e^S + B)$
    if $|\mathcal{O}| = 0$ **then** ▷ No overlap
      $\mathcal{M}_S \leftarrow$ set of mapped read positions in S
      $\mathcal{C}_a \leftarrow \mathcal{C}_a \cup (\mathcal{C}_P \cap \mathcal{M}_S)$
      $T.\mathsf{addInterval}(p_s^S, p_e^S)$
  return $\frac{|\mathcal{C}_a|}{|r|}$

D Time and memory

	Time (CPU h.)		Peak RAM (GB)	
	Min	Max	Min	Max
Ratatosk	11,698	22,252	213.22	305.29
FMLRC	4,228	10,828	122.69	253.49

The reported running times do not include data preprocessing for both tools which account for a negligible amount of time and memory compared to the total running time. On average, Ratatosk is 3.13 times slower than FMLRC. However, preliminary results on the HG002-1 data set suggests that Ratatosk v0.2 is 2.57 times faster than Ratatosk v0.1 which was used in our experiments.

Ratatosk was run in parallel on several machines due to the reference-guided preprocessing of the input data while FMLRC was run on a single machine. The reported peak of memory for Ratatosk matches the correction of the ambiguous LRS bin at the end of the preprocessing pipeline as it requires to use all the input SRS data. Indeed, Ratatosk memory usage is dominated by the graph index and the SRS data coloring of its vertices during the first correction pass.

E Tools

Tool	Version / Commit	URL
Ratatosk	0.1	https://github.com/DecodeGenetics/Ratatosk
FMLRC	77dde49	https://github.com/holtjma/fmlrc
minimap2	2.14-r883	https://github.com/lh3/minimap2
Clair	2.0.6	https://github.com/HKU-BAL/Clair
rtg	3.10.1	https://github.com/RealTimeGenomics/rtg-tools
Flye	2.7.1	https://github.com/fenderglass/Flye
QUAST	5.0.2	https://github.com/ablab/quast
Merqury	ed5918c	https://github.com/marbl/merqury

Script quast_sv_extractor.py was obtained from HELEN (https://github.com/kishwarshafin/helen/tree/master/helen/modules/python/helper).

F HG002 Data

Data	URL
ONT Guppy 3.6	https://precision.fda.gov/challenges/10
ONT Guppy 3.3	
Illumina	https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/data/AshkenazimTrio/HG002_NA24385_son/NIST_HiSeq_HG002_Homogeneity-10953946/NHGRI_Illumina300X_AJtrio_novoalign_bams
Small variants	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/AshkenazimTrio/HG002_NA24385_son/NISTv3.3.2/GRCh38
SVs*	ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/AshkenazimTrio/HG002_NA24385_son/NIST_SV_v0.6/
HiFi + HiCanu	https://obj.umiacs.umd.edu/marbl_publications/hicanu/hg002_hifi_hicanu_combined.fasta.gz
Ash1 v1.7	ftp://ftp.ccb.jhu.edu/pub/data/Homo_sapiens/Ash1/v1.7/Assembly/

* Require to be adapted from GRCh37 to GRCh38 by changing the contig names and lifting over the genomic regions to run with quast_sv_extractor.py.