Symphonizing pileup and full-alignment for deep learning-based long-read variant calling

Zhenxian Zheng, Shumin Li, Junhao Su, Amy Wing-Sze Leung, Tak-Wah Lam, Ruibang Luo*

Department of Computer Science, The University of Hong Kong, Hong Kong, China

* Email: rbluo@cs.hku.hk

Abstract

Deep learning-based variant callers are becoming the standard and have achieved superior SNP calling performance using long reads. In this paper, we present Clair3, which makes the best of two major method categories: pile-up calling handles most variant candidates with speed, and full-alignment tackles complicated candidates to maximize precision and recall. Clair3 ran faster than any of the other state-of-the-art variant callers and performed the best, especially at lower coverage.

Maintext

The first preprint of DeepVariant[1] was released in late 2016, marking the beginning of the use of deep learning-based methods (DL methods) instead of traditional statistical methods for variant calling. Over the years, several DL methods have been developed. We are now witnessing a complete take-over, led by DeepVariant for short-read variant calling. Long-read variant calling, using Oxford Nanopore (ONT) data, on the other hand, has been dominated by DL-methods since the beginning, primarily owing to the difficulty caused by its higher base error rate in general. Although the DL methods for short-read and long-read have a lot in common, the problem of long-read variant calling is considered more difficult. This led to two major designs – using pileup or full-alignment as the input of the decision-making neural network – which are significantly different in both performance and speed. Long-read variant callers, including Clairvoyante[2], Clair[3], and Nanocaller[4], are pileup-based, in which the read alignments are summarized into features and counts before being inputted into a variant calling network. PEPPER-Margin-DeepVariant[5] (PEPPER) is full alignment-based. The input to the DeepVariant variant calling network is kept with spatial information in the read alignments and is tens of times larger than the pileup inputs in terms of size. Medaka[6] is consensus-based; it uses pileup input to generate a diploid consensus in the first iteration and two haploid consensuses in the second. The differences between the reference and consensuses are identified and combined into variants. These are all state-of-the-art algorithms; the pileup-based algorithms are usually superior in terms of time efficiency and the full-alignment algorithms provide the best precision and recall. However, while the two designs are not mutually exclusive, there have not been any studies combining pileup calling and full-alignment calling.

To fill the gap, we developed Clair3, the successor to Clair, which makes the best of both designs. It runs as fast as the pileup-based callers and performs as well as the full alignment-based callers. **Supplementary Figure 1** shows the workflow for Clair3. The philosophy behind Clair3 is to trust the full-alignment model unless the pileup model can make a quick but reliable decision. First, the pileup calling network goes through all the variant candidates that passed a coverage threshold and an alternative allele frequency threshold. Next, the high-quality pileup calls are used to phase the alignments and as part of the final output. Then, the alignments phased by WhatsHap[7] are used to generate full-alignment input that is ~23 times larger in size than the pileup input for each low-quality pileup call for full-alignment calling. Finally, the full-alignment calls are integrated with the high-quality pileup calls as the final output. More details and parameters about the Clair3 workflow, input/output, and network architecture are provided in **Methods**.

We benchmarked Clair3 against PEPPER (the current top-performing long-read variant caller), Medaka (ONT’s in-house developed variant caller), Longshot[8] (non-deep learning-based; works only with SNP), and Clair (the Clair3 predecessor) on two GIAB[9, 10] samples: HG003 and HG004. HG003 was tested on models (including a pileup and a full-alignment model) trained on HG001, 2, 4 and 5. HG004 was tested on models trained on HG001, 2, 3 and 5. The model availability and training details are in **Methods**. We chose to use ONT data base-called using Guppy 4 (version 4.2.2) for two reasons: 1) compared to the Guppy 5, which was released in mid-2021, Guppy 4's read accuracy is ~1.8% lower[11], which is more challenging to variant calling, so it can better test the speed and performance of different variant calling methods, and 2) as at the completion date of this paper, Guppy 4 base-called reads were still the latest version available for download by the Human Pangenome Reference Consortium[12]. A summary of the datasets used for training and testing is shown in **Supplementary Table 1**. The correct PEPPER and Medaka model for Guppy 4 data was chosen for benchmarking. The links to the dataset, and the versions, commands and parameters used for each tool are available in the **Supplementary Notes**.

The benchmarking results at coverage from 10x to 90x are shown in **Figure 1a**, **Supplementary Table 2**, and **Supplementary Table 3**. The observations of different tools on HG003 and HG004 are almost identical, ruling out the possibility of any tools' overfitting to a particular sample. In terms of the SNP F1-score, Clair3 outperformed the other tools at lower coverage (10x to 30x) and performed similar to PEPPER above 30x. above 50x, the SNP F1-score improvement became more subtle. However, the Indel F1-score kept increasing with coverage, although it slowed down above 50x. Looking at the precision and recall at 50x (**Figure 1b**), in terms of SNP, Clair3 achieved 99.67% and 99.60%, which is similar to PEPPER's 99.61% and 99.63%. In terms of Indel, Clair3 achieved 90.86% and 64.73%, higher than PEPPER's 87.62% and 57.42%. In terms of speed (**Figure 1c**), Clair3 and Clair ran the fastest (~8 hours), and PEPPER was second-fastest (~30 hours). We then compared Clair3 to PEPPER using the CMRG v1 small variant benchmarking dataset[13], which covers repetitive and highly polymorphic medically relevant genes, so it is more challenging than using GIAB. However, CMRG v1 is based on HG002. To ensure no testing variant was involved in training, instead of training a new model with HG002 left out, we selectively benchmarked the 5,837 (out of 21,232) small variants that are in CMRG v1, but not GIAB HG002. The results are shown in **Supplementary Figure 2** and **Supplementary Table 4**. Similar to the trends observed for HG003 and HG004, Clair3 outperformed PEPPER at 10x to 30x on SNP, and had a similar performance above 30x. We compared Clair3 to PEPPER by different genomic contexts according to the GIAB genome stratifications[14] v2.0 on HG003 at 50x. The results are shown in **Supplementary Figure 3** and **Supplementary Table 5**. In SNPs, Clair3 outperformed PEPPER on precision in low complexity and functional regions, but not in low mappability and segmental duplication regions. Clair3 and PEPPER had the same recall in different regions. In Indels, Clair3 outperformed PEPPER in both precision and recall in all regions.

The success of the Clair3 method lies in the effective distinction between true and false calls during pileup calling, so that only necessary candidates are sent to the much more computationally intensive full-alignment calling. **Figure 2a** shows that an effective distinction was achieved using variant quality. Using HG003 at 50x as an example, most false variant calls and false reference calls had a quality between 0 to 10, while the true calls were between 15 to 30. In reality, while the correctness of a pileup call is not known in advance, we empirically decided to send the bottom 30% of the pileup variant calls and the bottom 10% of the pileup reference calls to full-alignment calling as the default settings of Clair3 (See **Methods**). In the previous example, quality cut-off 16 was chosen for the variant calls, which resulted in 98% of the false variant calls and only 9% of the true variant calls being sent to full-alignment calling. Similarly, cut-off 19 was chosen for the reference calls, so that 98% of the false reference calls and only 11% of the true reference calls were sent to full-alignment calling. **Figure 2b** shows that ~62% of the pileup failed variant calls and ~31% of the pileup failed reference calls were correctly called in full-alignment calling. We tested sending different percentages of pileup variant calls to full-alignment calling, from 0% (pileup calling only) to 100% (full-alignment calling only). The results are shown in **Figure 2c** and **Supplementary Table 6**. Clair3's default, which had a similar performance to full-alignment calling but ran ~4 times faster, showed that integrating pileup and full-alignment calling is a better strategy than relying on only one of them.

The benchmarks focused on the more challenging ONT data, but the Clair3 method is not restricted to a certain sequencing technology. It should work particularly well in terms of both runtime and performance on noisy data. Clair3 was released six months ago and is currently in its ninth revision, having integrated plenty of feedback from the community and ONT. We observed in PEPPER's most recent update (r0.7 on Dec 22nd, 2021) that a module in the front of the pipeline that was used solely for variant candidate selection was repurposed to output summary-based variant calls to relieve the heavy full-alignment calling workload. We expect integrating pileup and full-alignment calling to be a common practice in deep learning-based variant calling in the future.

Method

The Clair3 workflow

As **Supplementary Figure 1** shows, pileup candidates that are above a coverage threshold and an allele frequency threshold are extracted, and then called using the pileup network. The pileup calls are grouped into variant calls (genotype 0/1, 1/1, and 1/2) and reference calls (0/0). Both groups are ranked according to variant quality (QUAL). High-quality heterozygous SNP calls (top 70% in 0/1 calls) are used in WhatsHap phasing to produce phased alignment for input to the full-alignment network. Low-quality pileup calls (defaulted to the lowest 30% of variants and 10% of reference calls) are then called again using the full-alignment network. Finally, the full-alignment calls and high-quality pileup calls are outputted. Clair3 supports both VCF and GVCF output formats.

Input/Output

Clair3 uses a pileup input design simplified from that of its predecessors, and a full-alignment input to cover as many details in the read alignments as possible. **Supplementary Figure 4** visualizes the pileup and full-alignment inputs of a random SNP, insertion, deletion, or non-variant. **The pileup input** is 594 integers – 33 genome positions wide with 18 features at each position – A+, C+, G+, T+, I_S+, I^1_S+, D_S+, D^1_S+, D_R+, A-, C-, G-, T-, I_S-, I^1_S-, D_S-, D^1_S-, and D_R-. A, C, G, T, I, D, +, - means the count of read support of the four nucleotides: insertion, deletion, positive strand, and negative strand. Superscript “^{1}” means only the indel with the highest read support is counted (i.e., all indels are counted if without “^{1}”). Subscript “$_S$”/“$_R$” means the starting/non-starting position of an indel. For example, a 3bp deletion with the most reads support will have the first deleted base counted in either D^1_S+ or D^1_S-, and the second and third deleted bases counted in either D_R+ or D_R-. The design was determined experimentally, but the rationale is that for 1bp indels that are easy to call, look into the differences between the “$_S$” counts, but reduce the quality if the “$_R$” counts and discrepancy between positions increase. **The pileup output** is the same as that for Clair, but short of the two indel length tasks. The indel allele (or two indel alleles) with the highest reads support is used as the output according to the decision made in the 21-genotype task. **The full-alignment input** is 23,496 integers – 8 channels of 33 genome positions and 89 maximum representable reads. The description of the eight channels is in the

Supplementary Note. **The full-alignment output** is the same as that of Clair. The two indel length tasks can represent the exact indel length from -15 to 15bp, or below -15bp/ above 15bp. An indel call with an exact length will output the most reads-supported allele at that length. Otherwise, the most reads-supported allele below -15bp/ above 15bp is outputted. In training, indel length task 1 is given the smaller number, and in all our variant calling experiments, no length predictions in task 1 larger than in task 2 were observed. **The maximum supported coverage** of pileup/full-alignment input was 144/89. Random subsampling was done on excessive coverage. If the coverage in a full-alignment input was below 89, the reads were centered.

Network architecture

The pileup and full-alignment networks are shown in **Supplementary Figure 5**. **The pileup network** uses two bidirectional long short-term memory (Bi-LSTM) layers with 128 and 160 LSTM units. Stacked LSTM layers enable the network to learn the characteristics of raw sequential signal from different aspects at each position, but without increasing memory capacity, which enables the network to converge faster. Compared to Clair, the transpose-split layer is removed for a 40% speedup with a small performance loss that is taken care of in full-alignment calling. **The full-alignment network** is residual neural network (ResNet) alike and uses three standard residual blocks. A convolutional layer is added on top of each residual block to expand channels but reduce dimensionality across channels. A spatial pyramid pooling[15] (SPP) layer is used to tackle the problem of varying coverage in full-alignment input. SPP is a pooling layer that removes a network's fixed-size constraint, thus avoiding the need for input cropping or warping at the beginning. The SPP layer generates various receptive fields using three pooling scales (1x1, 2x2, and 3x3) in each channel. It then pools the receptive fields of all channels and generates a fixed-length output for the next layer. In both networks, the dropout rates of 0.2 for the flatten layer, 0.5 for the penultimate dense layer, and 0.2 for the task-specific final dense layers, are empirically determined.

Model availability and training

Pretrained models are provided in Clair3's installation. Models for specific chemistries and basecallers that are tested and supported by the ONT developers are available through Rerio (https://github.com/nanoporetech/rerio). The detailed steps, options and caveats for training a pileup model and a full-alignment model are available in Clair3's GitHub repo and are continually updated. The pretrained models, while targeted for use in production, were trained using multiple GIAB samples with known variants and 10 coverages for each sample, as described in Clair, but they always hold out chromosome 20 in Clair3. We used the following new training technics in Clair3. **(1) Representation Unification**: a variant can be represented in multiple forms[14]. Traditional variant calling methods rely on postprocessing (e.g., hap.py, RTG Tools) to equate multiple forms. However, to generate correct training samples, Clair3 must unify a variant's representations between the alignments and the truth variants. **Supplementary Figure 6** shows four cases in which the alignments and the truth variants have different representations that would confuse the training if not unified. Clair3 chooses to align the truth variants' representation to the alignments. The five detailed steps are shown in **Supplementary Figure 7**. First, the truth variants and alignments are phased (if not yet done) using WhatsHap. Second, among the candidates with alternative allele frequency ≥0.15, confident and *in situ* matches between the alignments and truth variants are identified and excluded from computationally intensive step 3. Third, the best match between the possible haplotypes of the truth variants and candidates is sought. Each of the truth variants can be either positive (using its reported genotype) or negative (using 0|0), and their Cartesian product forms possible haplotypes of the truth variants. Similarly, each candidate can be either 0|0, 0|1 (or 1|0 according to the phased alignments), or 1|1, and their Cartesian product forms the possible haplotypes of the candidates. A pairwise comparison is then done to find equivalent haplotypes between the two Cartesian products, and among all equivalents, the candidate haplotype with the most reads support is selected. The variants in the haplotype are used as the new truth variants. This step is computationally intensive, so in practice, we applied the step to partitions with at most 15 candidates and required less than 100bp between the candidates. Fourth, low alternative allele frequency (≥0.08 but <0.15) candidates with *in situ* matches between the alignments and the truth variants were chosen. Fifth, the truth variants or unified variants generated in steps 2, 3 and 4 were merged. In our benchmarks, representation unification alone in general increased the SNP recall by ~0.2% and Indel recall by ~2%. **(2) Ratio of variants to non-variants samples for training**: In Clair, the ratio was fixed at 1:2. In Clair3, we tested ratios up to 1:10 for both pileup and full-alignment model training, and we observed a monotonic but decelerated performance increase with more non-variants added to the training. Since focal loss is used to alleviate the effect of training class imbalance, another possible explanation is that the 21-genotype output task that Clair3 relies primarily on is insensitive to the ratio because it judges only the genotype of a candidate instead of whether a candidate is a variant or not. We chose 1:5 and 1:1 as the default ratio for pileup and full-alignment model training, respectively, to strike a balance between model performance and training speed. **(3) Use of phased alignments**: Deep-learning and full-alignment based variant callers DeepVariant and PEPPER concluded that using phased alignments is essential to their high performance. In Clair3, high-quality heterozygous pileup calls are used to phase the input alignments using the 'phase' and 'haplotag' modules in WhatsHap. The phased alignments are used as input for full-alignment calling. When training a full-alignment model, two training samples for each variant, one using phased alignments and the other unphased, are used to ensure the model works when alignments cannot be properly phased. In our benchmarks, the use of phased alignments alone, in general, increased the SNP F1-score by ~0.1%, and the Indel F1-score by ~6%. **(4) New optimization methods**: Clair3 removed both the cyclical learning rate and learning rate decay strategies used in Clair, and now uses the Ranger optimizer (RectifiedAdam[16] plus Lookahead[17]) for automated warm-up, faster convergence, minimal computational overhead, etc. In our benchmarks, compared to Clair, the new optimizer alone, in general, increased the overall F1-score of pileup calling by ~0.2%.

Benchmarking methods and computational concerns

We used five GIAB samples, HG001 to 5, for either model training or testing. When using either HG003 or HG004 for testing, the other four samples were used for training. We selected 10% of the training samples for validation and chose the best-performing epoch in the first 30 epochs in the validation data for benchmarking. We used hap.py[14] to compare the called variants against the true variants, and used Clair3's 'GetOverallMetrics' module to generate three metrics, 'precision', 'recall', and 'F1-score', for five categories: 'overall', 'SNP', 'Indel', 'Insertion', and 'Deletion'. We used qfy.py with V2.0 GIAB genome stratifications to evaluate Clair3's performance in challenging and targeted regions of the genome. Runtimes were gauged on a server with two 2.1GHz Intel Xeon Silver 4116s, with 24 cores, and 256GB memory at 2666MHz. With the same setting, Clair3 finished in ~8 hours using ~50x of ONT Guppy 4 data and in ~4.5 hours with the same amount of Guppy 5 data. The memory consumption of each Clair3 calling process was capped at 1GB.

Brief summary of methods tested showing no or negligible improvement

(1) Use of more residual blocks in the full-alignment network: We added a fourth residual block with 512 channels. The number of parameters increased from 2,989,210 to 9,812,634. The runtime doubled, but the performance change was negligible, even though the terminal training loss fell. **(2) Local realignment**: This technique is essential for high indel calling performance in state-of-the-art, short-read, small variant callers. But it worked differently on long-read. We tried local realignment using a 2000bp window in regions with a high density of candidates using a local realignment algorithm similar to that of DeepVariant. We observed that while it increased the recall a bit, local realignment tripled the runtime and introduced ~10% of new non-variant candidates, which in turn, lowered the precision a bit. In Clair3, we implemented local realignment, but disabled it on long-read as the default. **(3) Including variants outside high-confidence regions in training**: To increase variant training samples, we explored including variants outside the high-confidence regions in training, but observed negative performance improvement in Clair. In Clair3, the GIAB truth datasets we used were upgraded from version 3.3.2 to 4.2.1, but we had the same observation that including variants outside the high-confidence regions in training jeopardized model performance. **(4) Selecting candidates for full-alignment calling based on reference sequence complexity**: Variant calling is more difficult in the "low complexity" and "difficult to map" regions. In addition to selecting candidates by pileup calling quality ranking for full-alignment calling, we added those candidates at positions with relatively low sequence entropy (the lowest 30% of the whole genome). About three times more candidates were selected for full-alignment calling, but the performance increase was negligible.

Code availability

Clair3 is open-source software (BSD 3-Clause license), hosted by GitHub at https://github.com/HKU-BAL/Clair3, and available through Docker, Bioconda, and Singularity.

Data availability

The 1) links to the reference genomes, truth variants, benchmarking materials, and ONT data, and 2) the commands and parameters used in this study, are available in the Supplementary Notes. All analysis output, including the VCFs and running logs, are available at http://www.bio8.cs.hku.hk/clair3/analysis_result.

Acknowledgements

R. L. was supported by Hong Kong Research Grants Council grants GRF (17113721), ECS (27204518), and TRS (T21-705/20-N and T12-703/19-R), and the URC fund at HKU.

Author contributions

R. L. conceived the study. Z. Z. and R. L. designed the algorithms, implemented Clair3, and wrote the paper. All authors evaluated the results and revised the manuscript.

Competing interests

R. L. receives research funding from ONT. The remaining authors declare no competing interests.

Figures

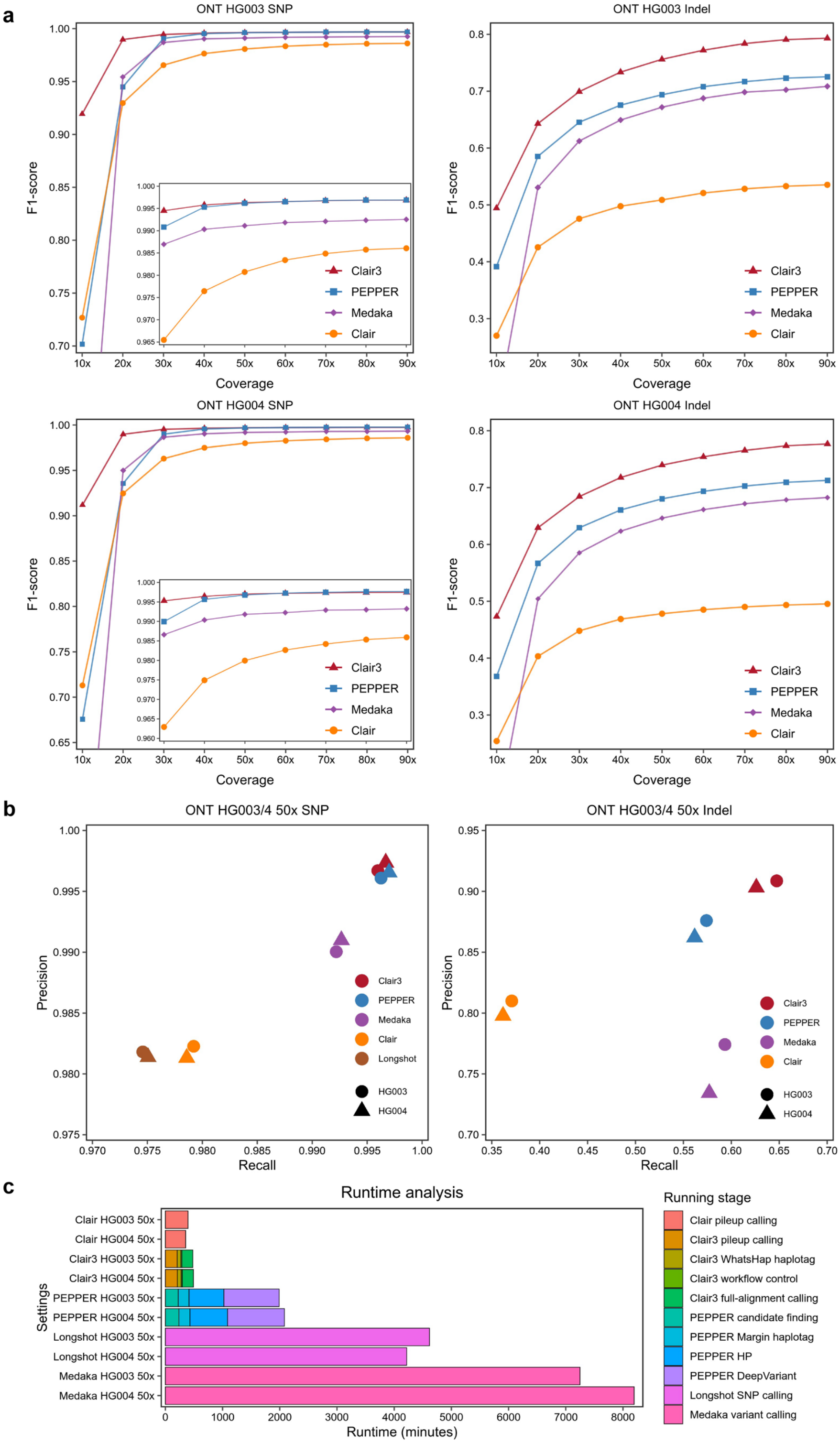

Figure 1. Benchmarking results on HG003 and HG004. (a) The SNP/Indel F1-score of different tools at multiple coverage from 10x to 90x. (b) The precision against the recall of different tools at 50x coverage. (c) The runtime breakdowns of different tools at 50x coverage.

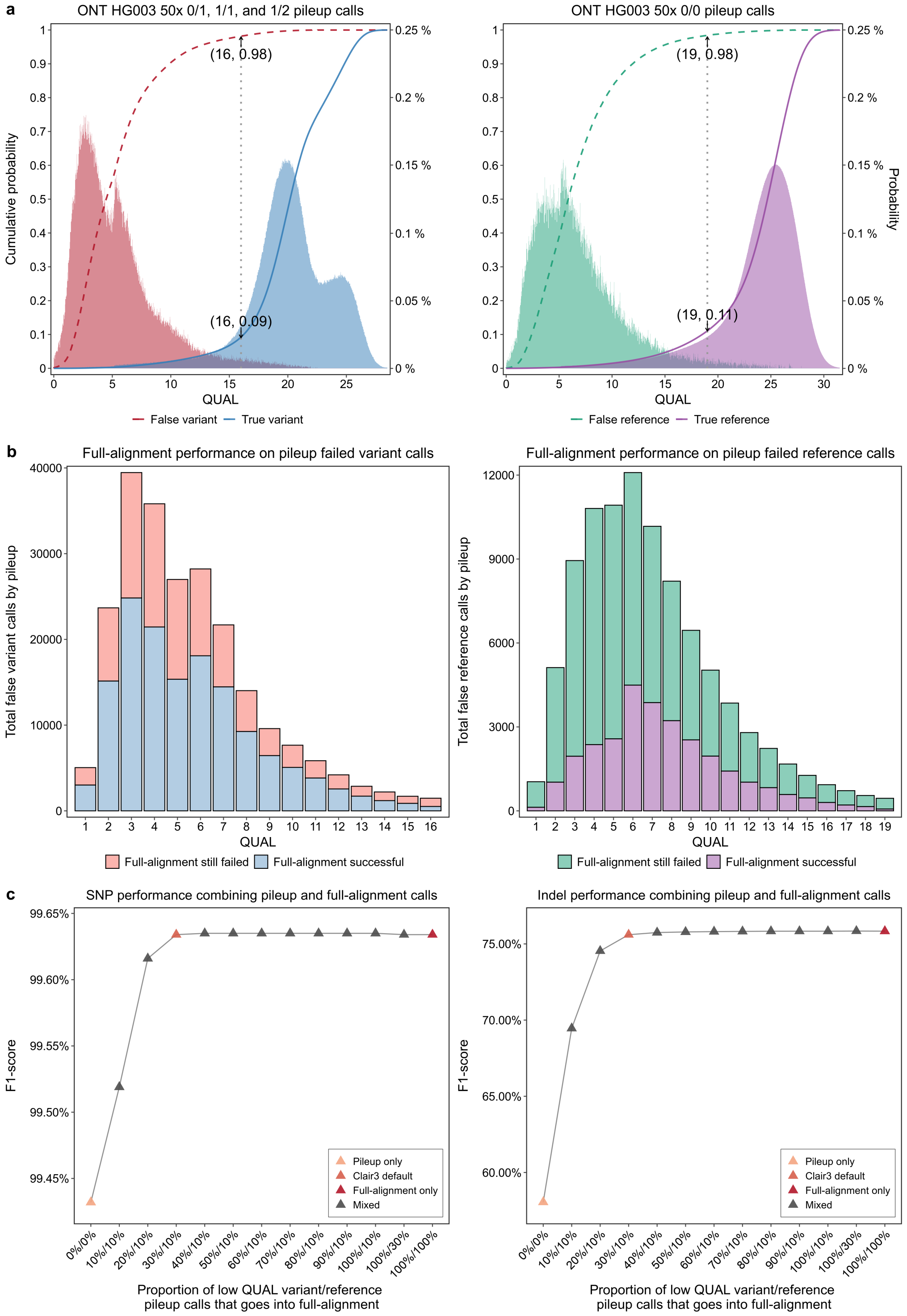

Figure 2. **Pileup and full-alignment calling working details and synergy on HG003 at 50x coverage.** (a) The variant quality distribution of the true and false variant/reference pileup calls. (b) The performance of full-alignment on pileup failed variants of different variant quality. (c) The F1-score when different proportions of low-quality variant/reference calls enter full-alignment calling.

References

1. Poplin, R. et al. A universal SNP and small-indel variant caller using deep neural networks. *Nature biotechnology* **36**, 983-987 (2018).
2. Luo, R., Sedlazeck, F.J., Lam, T.-W. & Schatz, M.C. A multi-task convolutional deep neural network for variant calling in single molecule sequencing. *Nature communications* **10**, 1-11 (2019).
3. Luo, R. et al. Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nature Machine Intelligence* **2**, 220-227 (2020).
4. Ahsan, M.U., Liu, Q., Fang, L. & Wang, K. NanoCaller for accurate detection of SNPs and indels in difficult-to-map regions from long-read sequencing by haplotype-aware deep neural networks. *Genome Biology* **22**, 261 (2021).
5. Shafin, K. et al. Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads. *Nature methods* **18**, 1322-1332 (2021).
6. Medaka, https://github.com/nanoporetech/medaka.
7. Patterson, M. et al. WhatsHap: weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology* **22**, 498-509 (2015).
8. Edge, P. & Bansal, V. Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing. *Nature communications* **10**, 1-10 (2019).
9. Olson, N.D. et al. precisionFDA Truth Challenge V2: Calling variants from short-and long-reads in difficult-to-map regions. *Biorxiv*, 2020.2011. 2013.380741 (2021).
10. Wagner, J. et al. Benchmarking challenging small variants with linked and long reads. *BioRxiv* (2020).
11. Nanopore EPI2ME Labs, https://labs.epi2me.io/gm24385_2021.05/.
12. Shafin, K. et al. Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nature biotechnology* **38**, 1044-1053 (2020).
13. Wagner, J. et al. Towards a Comprehensive Variation Benchmark for Challenging Medically-Relevant Autosomal Genes. *bioRxiv* (2021).
14. Krusche, P. et al. Best practices for benchmarking germline small-variant calls in human genomes. *Nature biotechnology* **37**, 555-560 (2019).
15. He, K., Zhang, X., Ren, S. & Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* **37**, 1904-1916 (2015).
16. Liu, L. et al. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265* (2019).
17. Zhang, M.R., Lucas, J., Hinton, G. & Ba, J. Lookahead optimizer: k steps forward, 1 step back. *arXiv preprint arXiv:1907.08610* (2019).