Supplementary Data of manuscript "Universal Denoising of Aligned Genomic Sequences"

Irena Fischer-Hwang, Mikel Hernaez, Idoia Ochoa
and Tsachy Weissman

In this document we explain in detail the parameter choice process for SAMdude, provide the commands for all the denoisers, and provide the genomics toolkit commands for the SNP calling pipeline.

1 SAMdude parameter choice

We begin with a description of the parameter choices for SAMdude: single-sided context length k and confidence probability threshold t_p. We end this section with a description of sequence data coverage requirements for SAMdude.

1.1 Choice of k

For discrete universal denoising using DUDE, the choice of k is linked to the input sequence length with one valid optimal single-sided context length k_n given as:

$$k_n = \lceil c \log_{\mathrm{M}} n \rceil$$

		Raw variant calls				VQSR filtered variant calls			
Chr	k	ΔC	ΔS [%]	ΔP [%]	ΔF	ΔC	ΔS [%]	ΔP [%]	ΔF
11	5	26	0.01	0.03	–	50	0.01	0.02	–
	6	-199	0.03	0.17	0.001	-80	0.03	0.11	0.001
20	5	59	0.04	0.07	–	59	0.04	0.06	–
	6	-19	0.01	0.10	–	18	0.01	0.05	–
	10	373	0.21	0.16	0.002	72	0.20	0.12	0.001

Table 1: Denoising results for different values of k, with changes (Δ), T.P. and F.P. calculated relative to the original file. C is the number of variants called for each condition, sensitivity $\mathrm{S} = \left(\frac{\mathrm{T.P.}}{\mathrm{T.P.+F.N.}}\right)$, precision $\mathrm{P} = \left(\frac{\mathrm{T.P.}}{\mathrm{T.P.+F.P.}}\right)$, and F-score $\mathrm{F} = \frac{1}{2}(\mathrm{S}+\mathrm{P})$. For S, P and F, positive Δ indicates improvement with respect to the original data, and horizontal lines indicate no change.

with $c < \frac{1}{2}$, noise-free sequence alphabet size M and noise-free sequence length n. In the genomic sequencing case, M= 4, and n ranges from approximately 51,000,000 up to 249,000,000. For these values, $k_n = 5$ or 6.

Table 1 shows the results of denoising extracted SAM files for chromosomes 11 and 20 NA12878 paired-end WGS dataset ERR262997 using $k = 5$ and 6, and confidence probability threshold $t_p = 0.9$. As expected, these values of k resulted in improvements in both recall and precision, but very little change in F-Score. In order to improve denoising performance, we tried larger k values in the hopes that larger k would ensure that context counts would be taken from the same pileup, but still allow potential information from misaligned or poorly-mapped reads. The denoising results for $k = 7$ are presented in the main text, and here we additionally show results for $k = 10$ for chromosome 20. Although the best denoising performance was obtained for $k = 10$, the potential number of contexts for which count vectors would need to be recorded became computationally prohibitive.

		Raw variant calls				VQSR filtered variant calls			
Chr	k	ΔC	ΔS [%]	ΔP [%]	ΔF	ΔC	ΔS [%]	ΔP [%]	ΔF
11	5	71	0.01	0.03	–	103	0.01	0.02	–
	6	-238	–	0.20	0.001	-112	–	0.12	0.001
20	5	79	0.05	0.05	–	92	0.05	0.03	
	6	-23	-0.01	0.10	–	20	-0.03	0.03	–

Table 2: Denoising results for various k with confidence threshold $t_p = 0.99$, with changes (Δ), T.P. and F.P. calculated relative to the original file. C is the number of variants called for each condition, sensitivity $\mathrm{S} = \left(\frac{\mathrm{T.P.}}{\mathrm{T.P.+F.N.}}\right)$, precision $\mathrm{P} = \left(\frac{\mathrm{T.P.}}{\mathrm{T.P.+F.P.}}\right)$, and F-score $\mathrm{F} = \frac{1}{2}(\mathrm{S}+\mathrm{P})$. For S, P and F, positive Δ indicates improvement with respect to the original data, and horizontal lines indicate no change.

1.2 Choice of confidence probability threshold t_p

The choice of confidence probability threshold is a reflection of confidence in the sequencer's base calling. Too low a threshold might prevent numerous noisy bases from being denoised, while too high a threshold might result in changes to bases with very high confidence calls. Table 2 shows the results for changes in variant calls with a very high confidence probability threshold of 0.99, corresponding to a quality score of 20. Based on these results, we chose to use a lower confidence probability threshold of $t_p = 0.9$ for our denoising tests.

1.3 Coverage of input data

Table 3 shows the results of denoising using SAMdude for various values of k on SAM files extracted from NA12878 paired-end WGS dataset ERR174324 corresponding to 15×-coverage. The lack of improvement, and slight negative

		Raw variant calls				VQSR filtered variant calls			
Chr	k	ΔC	ΔS [%]	ΔP [%]	ΔF	ΔC	ΔS [%]	ΔP [%]	ΔF
11	5	-13	0.01	–	–	-7	0.01	–	–
	6	-209	–	0.01	–	-220	-0.02	0.01	–
	7	-451	-0.03	0.03	–	-382	-0.03	–	–
20	5	18	–	-0.01	–	39	0.01	–	–
	6	-111	-0.03	0.01	–	-103	-0.04	–	–
	7	-153	-0.02	–	–	-117	-0.03	–	–
	10	235	0.05	-0.04	–	189	0.03	-0.02	–

Table 3: Denoising results for low coverage data with confidence threshold $t_p = 0.9$, with changes (Δ), T.P. and F.P. calculated relative to the original file. C is the number of variants called for each condition, sensitivity S = $\left(\frac{\text{T.P.}}{\text{T.P.+F.N.}}\right)$, precision P = $\left(\frac{\text{T.P.}}{\text{T.P.+F.P.}}\right)$, and F-score F = $\frac{1}{2}$(S + P). For S, P and F, positive Δ indicates improvement with respect to the original data, and horizontal lines indicate no change.

effect of SAMdude denoising on these lower coverage data are not entirely surprising. SAMdude makes significant use of alignment information in order to estimate the noise channel, create counts vectors and to perform denoising, so we expect better performance with more coverage. Based on these results, we focused our efforts on denoising data with 30× or higher coverage, which tends to be the coverage range of WGS data used in practice, especially for clinical purposes.

2 SNP calling pipeline

In this section we describe the steps and pipelines used to analyze the effect of base denoising and quality score updating on SNP calling.

2.1 Preprocessing

The first steps in the pipeline are preprocessing steps which include conversion of the SAM file to the BAM format, file sorting, duplicate reads marking, group name adding, file indexing and quality score recalibration.

The SAM file is converted to the BAM format using samtools, specifying the number of threads with the -@ option and inclusion of the SAM header with the -h option.

```
$ samtools view -@ num_threads -b -h aln.sam > aln.bam
```

The BAM file is then sorted using samtools, with a temporary file prefix specified with the -T option, and the output file format specified with the -O option.

```
$ samtools sort -T ./tmp -@ num_threads -O bam aln.bam > aln.sorted.bam
```

Duplicates are marked using Picard tools [1], with M specifying the file to which metrics calculated during the duplication marking process are written. Note that marking the duplicates is sufficient to exclude them from downstream processes.

```
$ java -jar picard.jar MarkDuplicates I=aln.sorted.bam \
      O=aln.sorted.dedup.bam M=metrics.txt ASSUME_SORTED=true
```

Read group names are then added to the file, with read group parameters specified by RGID, RGLB, RGPL, RGPU and RGSM.

```
$ java -jar picard.jar AddOrReplaceReadGroups INPUT=aln.sorted.dedup.bam \
      OUTPUT=aln.sorted.dedup.rg.bam RGID=group1 RGLB=lib1 \
      RGPL=illumina RGPU=unit1 RGSM=NA12878
```

Then, the BAM file is indexed.

```
$ java -jar picard.jar BuildBamIndex I=$aln.sorted.dedup.rg.BAM
```

Finally, the quality scores are recalibrated using the Genome Analysis Toolkit [1] Base Quality Score Recalibration workflow, and the NCBI build 37 of the Human reference[1] as the reference genome.

```
$ java -jar GenomeAnalysisTK.jar -nct num_threads -T BaseRecalibrator -R pa
      -I aln.sorted.dedup.rg.bam \
      -knownSites bundle_2.8/dbsnp_138.b37.vcf \
      -knownSites bundle_2.8/Mills_and_1000G_gold_standard.indels.b37.vcf \
      -knownSites bundle_2.8/1000G_phase1.indels.b37.vcf -o bqsr.data
```

```
$ java -jar GenomeAnalysisTK.jar -nct num_threads -T PrintReads -R pathHuma
      -I aln.sorted.dedup.rg.bam -BQSR recal_data -o aln.sorted.dedup.rg.re
```

2.2 Variant Calling and filtering

We use the GATK Haplotype Caller for variant calling, specifying the target chromosome with the -L option.

```
$ java -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R pathHumanReference \
      -I aln.sorted.dedup.rg.recal.bam -L targetRegion \
      --genotyping_mode DISCOVERY -stand_emit_conf 10 -stand_call_conf 30 -
```

Finally, the Hap.py evaluation pipeline is used to filter variants and extract true and false positives.

```
$ python hap.py ground_truth.vcf $raw_VCF -f ground_truth.bed -o results -r
$ python rep.py -o results.html -l tsv_file
```

[1] http://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13/

References

[1] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly *et al.*, "The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data," *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.