Sep 30, 2024 Version 1

Bioinformatic pipeline for analysing variations in long-reads reconstructed human genomes V.1

DOI
dx.doi.org/10.17504/protocols.io.36wgqnoz3gk5/v1

Angelo Sante Varvara[1], Ilenia Urso[2], Sharon Natasha Cox[1], Graziano Pesole[1,2]

[1]Department of Biosciences, Biotechnology and Environment, University of Bari Aldo Moro, Bari, Italy;
[2]Institute of Biomembranes, Bioenergetics and Molecular Biotechnologies, Consiglio Nazionale delle Ricerche, Bari, Italy

Human_ONT

Angelo Sante Varvara
University of Bari

OPEN ACCESS

DOI: dx.doi.org/10.17504/protocols.io.36wgqnoz3gk5/v1

Protocol Citation: Angelo Sante Varvara, Ilenia Urso, Sharon Natasha Cox, Graziano Pesole 2024. Bioinformatic pipeline for analysing variations in long-reads reconstructed human genomes. **protocols.io https://dx.doi.org/10.17504/protocols.io.36wgqnoz3gk5/v1**

Protocol status: Working
We use this protocol and it's working

Created: July 08, 2024

Last Modified: September 30, 2024

Protocol Integer ID: 102990

Keywords: Bioinformatics, ONT, Nanopore, variants, human, variations, reads, indels, SNV, SV, alignment, phasing, dorado, clair3, whatshap, modkit, modified bases, variant calling, IGV

Abstract

Recent advances in sequencing technologies have paved the way for personalized medicine and clinical genomics, providing unprecedented insights into the genetic basis of human health and disease. High-throughput sequencing is now recognized as an effective diagnostic tool for patients affected by rare diseases but the use of either Whole Exome Sequencing (WES) or Whole Genome Sequencing (WGS) technologies has shown to be uninformative in many cases. On the contrary, the recent introduction of long-read technologies can resolve genomic regions inaccessible to short-read sequencing, but a reliable benchmark assessment is missing.

Following a long and intense benchmarking activity, here we present our bioinformatic pipeline based on Oxford Nanopore Technology (ONT) long reads aimed at shading lights to the stages of alignment, SNVs, indels, and SV calling, and modified bases analysis.

Attachments

protocol.jpg

337KB

Before start

#Versions of tools used in this pipeline:

- Dorado 0.7.2
- samtools 1.19.2
- NanoStat 1.6.0
- pycoQC 2.5.2
- Clair3 1.0.9
- Clair3-Nova 0.3.0
- bcftools 1.19
- sniffles2 2.2
- sniffles2_plots 0.2.0
- SvAnna 1.0.4
- whatshap 2.1
- modkit 0.2.2

Please kindly note that whilst running Dorado without a GPU is technically possible, it is strongly inadvisable as basecalling will be much slower when running purely on CPU.
Dorado is heavily optimized for Nvidia A100 (ampere) and H100 (hopper) GPUs and will deliver maximum performance on nodes containing these GPUs.

Basecalling and alignment

11h

1 Basecalling raw data in pod5 format with D*orado basecalling.*

9h

Dorado is a high-performance open-source basecaller for Oxford Nanopore reads.
First, let's download the latest models for our needs with *Dorado download*.

Command

Downloading Dorado models

```
dorado download --model dna_r10.4.1_e8.2_400bps_sup@v5.0.0

dorado download --model
dna_r10.4.1_e8.2_400bps_sup@v5.0.0_5mCG_5hmCG@v1
```

Then, we can run D*orado basecaller* on our raw data in pod5.
If your raw data are in fast5 format, it is recommended to convert them in pod5 with *pod5 convert fast5* tool to reach optimal performances.
For our test, we are going to use human Genome assembly GRCh38 as a reference, calling both 5mC and 5hmC modified bases, using the most updated and accurate models.

Command

Basecalling with Dorado

```
dorado basecaller dna_r10.4.1_e8.2_400bps_sup@v5.0.0 /sample/pod5/ --
min-qscore 10 --recursive --modified-bases-models
dna_r10.4.1_e8.2_400bps_sup@v5.0.0_5mCG_5hmCG@v1
--reference hg38.fa > calls.bam
```

Following the basecalling step, create a sequencing summary with D*orado summary, it* outputs a tab-separated file with read-level sequencing information from the BAM file generated during basecalling.

Command

Running Dorado summary

```
dorado summary calls.bam > sequencing_summary_dorado.txt
```

2 **Sort and index the resulting alignment with *samtools*.** 2h

It's time to sort the alignment in bam format with *samtools sort*.

Command

Sorting alignment

```
samtools sort -@ 16 calls.bam -o calls_sorted.bam
```

Indexing the sorted bam is mandatory for further steps.

Command

Indexing alignment

```
samtools index -@ 16 calls_sorted.bam
```

Alignment statistics 1h

3 **Create alignment statistics with *NanoStat, samtools,* and *pycoQC*.** 1h

NanoStat highlights alignment information as, for instance, N50 or read length and read quality.

Command

Running NanoStat

```
NanoStat --summary sequencing_summary_dorado.txt -t 12 > Nanostats.txt
```

Samtools utilities give us insights on coverage and mapping values.

Command

Running samtools stats and samtools coverage

```
samtools stats -@ 16 calls_sorted.bam > samtools_stats.txt

samtools coverage calls_sorted.bam > samtools_coverage.txt
```

PycoQC produces an html format file with tons of charts regarding basecalling progress and outcomes.

Command

Running pycoQC

```
pycoQC -f sequencing_summary_dorado.txt -o report_dorado.html
```

Variant calling

2h 35m

4 SNV and indels calling with C*lair3*.

2h

Variant calling is performed with C*lair3* and we strongly advise these parameters, with the same model path used for the basecalling stage.

Command

Running clair3

```
run_clair3.sh --bam_fn="calls_sorted.bam" --ref_fn="hg38.fa" --
threads=16 --platform="ont" --include_all_ctgs --
model_path="dna_r10.4.1_e8.2_400bps_sup@v5.0.0" --output="clair3"
```

5 Polishing calls with *bcftools*.

10m

This step depends on your alignment statistics. We usually advise removing all "lowQual" calls as well as variants with depth below 5 (or 10 if your coverage is above 30x).

Command

Filtering vcf

```
bcftools view merge_output.vcf.gz -i 'FILTER="PASS" & FMT/DP>5' --
threads 8 --output filtered_1.vcf.gz
```

Furthermore, *bcftools norm* command will split all multi-allelic variants into multiple rows and furtherly check calls with the reference throwing out redundancies or errors, producing a vcf file with improved accuracy.

Command

Running bcftools norm

```
bcftools norm -m-any --check-ref -w -f hg38.fa filtered_1.vcf.gz -o
filtered_2.vcf.gz
```

Lastly, it can probably be helpful to add further info to the final vcf file, especially in pathogenic studies. Annovar and SnpEff are both powerful tools that can fill variant calls with annotations, as those from ClinVar, Gnomad and dbSNP.

6 **SV calling with *sniffles2*.** 10m

We strongly advise using the bed files found in *sniffles* GitHub repository to achieve better performances while calling structural variants in repetitive regions.

Command

Running sniffles2

```
sniffles --input calls_sorted.bam --vcf sniffles.vcf --snf
sniffles.snf --reference hg38.fa --minsvlen 50 --output-rnames --
threads 16 --tandem-repeats human_GRCh38_no_alt_analysis_set.trf.bed
```

Calls can be subsequently filtered and polished depending on our needs with *bcftools*.

6.1 Additionally, *sniffles2* developers offer a python program that generates a wide range of plots for single and multi-sample VCF files, such as SV size & type distribution, and genotype frequency. 5m

Command

Creating plots

```
python3 -m sniffles2_plot -i sniffles.vcf -o sniffles2_plots
```

7 **SV annotation and prioritization with *SvAnna.***

10m

Structural variants need to be annotated as well as, in case of pathogenic analysis, prioritized. We benchmarked several tools for this purpose, but we determined SvAnna to be the most suitable for our goals, taking as inputs either *sniffles2* vcf and our HPOs.
The output in html is specifically useful in order to have a first look at the most valuable structural variants in terms of pathogenic relevance.

Command

Running SvAnna

```
java -jar svanna-cli-1.0.4.jar prioritize -d /svanna-cli-1.0.4/svanna-data --output-format html,vcf --vcf sniffles.vcf --phenotype-term HP:0001250 --phenotype-term HP:0001249 --phenotype-term HP:0000256 --n-threads 8 --out-dir /svanna-cli-1.0.4/output
```

Phasing

3h 35m

8 **Phasing heterozygous variants with *whatshap phase.***

30m

This tool allows phasing variants (called via clair3), annotating the info directly in our vcf file.

Command

Running whatshap phase

```
whatshap phase -o phased.vcf.gz --reference hg38.fa --ignore-read-groups merge_output.vcf.gz calls_sorted.bam
```

Please note that also structural variants can be phased with *whatshap* phase in the same way.

9 **Phasing assessment with *whatshap stats.*** 5m

Assessing the amount of phased heterozygous variants is crucial, and we can achieve this with *whatshap stats*. It will output various statistics which can vary a bit depending on the alignment quality (coverage, N50, etc), but we can safely conclude that a percentage above 90% is expected in 30X covered genomes.

Command

Running whatshap stats

```
whatshap stats --tsv=whatshap.tsv --gtf=whatshap.gtf phased.vcf.gz > stats_whatshap.txt
```

10 **Tagging reads by haplotype with *whatshap haplotag*.** 3h

Quite often it is interesting to show the reads as well as the variants. For that, we can run the *whatshap haplotag* subcommand on the phased VCF file. It tags each read in a BAM file with HP:i:1 or HP:i:2 depending on which haplotype it belongs to, and also adds a PS tag that describes in which haplotype block the read is.
The command below creates a BAM file 'haplotagged.bam' with the tagged reads, which you can open in IGV, in order to visualize either haplotype assignment and modified bases.

Command

Running whatshap haplotag

```
whatshap haplotag -o haplotagged.bam --reference hg38.fa
phased.vcf.gz calls_sorted.bam
```

Tips for trios

11 In case we had a trio of samples (child, parent1, and parent2) the above stages of variant calling and phasing can be completed with a single command thanks to the Clair3-Nova tool. With few lines of code, we would be able to perform SNVs and indels calling, phasing heterozygous variants, and haplotagging input bam files with haplotype infos, the two latter steps ran via whatshap if we specify the tool path.
If needed, Clair3-Nova can output also gvcfs files.

12h

Command

Running Clair3-Nova

```
run_clair3_nova.sh \
  --bam_fn_c=child_sorted.bam \
  --bam_fn_p1=P1_sorted.bam \
  --bam_fn_p2=P2_sorted.bam \
  --output=clair3_nova \
  --ref_fn=hg38.fa \
  --threads=12 \
  --model_path_clair3=r1041_e82_400bps_sup_v430 \
  --model_path_clair3_nova=r1041_e82_400bps_sup_nova \
  --gvcf \
  --include_all_ctgs \
  --whatshap=/usr/bin/whatshap \
  --enable_output_phasing \
  --enable_output_haplotagging
```

Modified bases

3h

12 **Modified bases calling and stats with *modkit*.**

3h

Modified bases can be called with *modkit* directly from the modBAM file, depending on the type of model chosen during the basecalling stage. Modkit can effectively count the following modified bases: 5mC, 5hmC, 6mA. It outputs a bedMethyl file, a table with the counts of base modifications from every sequencing read over each reference genomic position. No reference sequence is required (hence we strongly advise using it) and for user convenience, the counting process can be modulated using several additional transforms and filters, for instance the most basic of these is to report only counts from reference CpG dinucleotides.

Command

Running modkit pileup

```
modkit pileup calls_sorted.bam pileup_m.bed --ref hg38.fa --log-
filepath pileup.log -t 16
```

It is also possible to produce a summary to have an overview of the mod tags that are present in a BAM and get basic statistics. The default output is a totals table (designated by '#' lines) and a modification calls table. We ran the command with *--no-sampling* flag to count modified bases throughout the genome but if you prefer, you can choose a specific region or a specific number of reads to be randomly picked.

Command

Running modkit summary

```
modkit summary calls_sorted.bam --no-sampling -t 16
```

Furthermore, modkit offers the possibility to perform differential methylation scoring. This can be done using two subcommands: *modkit dmr pair* to compare regions in a pair of samples (for example, tumor and normal, or control and experiment), or *modkit dmr multi* to compare regions between all pairs of samples (for example, a trio sample set). Before performing this step, the input data (i.e., bedMethyl files created in the previous step) must be compressed with bgzip and indexed with tabix. Additionally, you must specify a BED file containing the specific regions to be compared, such as CpG Islands in hg38. This BED file should contain 4 columns: chrom, chromStart, chromEnd, and CpGname. Keeping the name column is optional and we can also specify the modified base in which we are interested in, for example, for CpG islands we can specify "--base C".

Command

Differential methylation scoring for a pair of samples

```
modkit dmr pair -a control.bed.gz --index-a control.bed.gz.tbi -b
case.bed.gz --index-b case.bed.gz.tbi -o
differential_case_control.bed -r CpG.bed --ref hg38.fa --base C --
threads 16 --log-filepath file.log
```

The *modkit dmr multi* command performs pairwise comparisons across all regions specified in the regions BED file for more than two samples. The data preparation is the same as described in the previous section, applied to each sample. An example command for a trio is the following:

Command

Differential methylation scoring on all pairs of samples

```
modkit dmr multi -s father.bed.gz norm1 --index father.bed.gz.tbi
norm1 -s mother.bed.gz norm2 --index mother.bed.gz.tbi norm2 -s
child.bed.gz child1 --index child.bed.gz.tbi child1 -o out_path --
regions-bed CpG.bed --ref hg38.fa --base C -t 32 --log-filepath
file.log
```

The latest version of modkit introduces the option "--segments", which allows *modkit dmr* to dynamically identify and generate regions of differential methylation during the analysis using a hidden Markov model, instead of relying on pre-defined regions. This can provide more flexible and potentially more accurate identification of methylation patterns in your data

Command

Differential methylation using a Hidden Markov Model with --segments option

```
modkit dmr pair --segment segmentation.bed  -a control.bed.gz --index-
a control.bed.gz.tbi -b case.bed.gz --index-b case.bed.gz.tbi -o
out_path --ref hg38.fa --base C -t 32 --log-filepath file.log
```

Further analysis

13 Further analyses are expected to be performed according to the specific aim of your study.