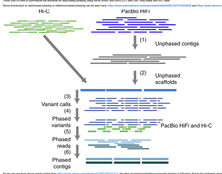
Read-based phasing or reference-based phasing!!!

Created by Chen Yutang, last modified just a moment ago

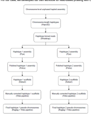
I think now it's time to summarize the workflow for read-based phasing using WHATSHAP and HAPCUT2 with ONT long reads and Hi-C data.

Some introduction to read-based phasing or reference-based phasing can be seen here, https://www.sciencedirect.com/science/article/pii/S2001037019303836 and https://www.nature.com/articles/s41587-020-0711-0.



As we can see from above graph copied from https://www.nature.com/articles/s41587-020-0711-0, the idea of read-based/reference-based phasing is following: first build unphased chromosomes, then use unphased chromosomes as reference, mapping short reads and long reads to the reference to derive chromosome-level phase blocks. Next, based on the chromosome-level phase block, long reads are split into two haplotype groups and finally each group of reads is assembled separately to reconstruct both homologous chromosomes. Any two SNPs in the chromosome covered by long reads will be phased, and many local phase blocks will be formed if there are no SNPs shared between long reads to extend the phase. Additional long-range linkage data, such as Hi-C reads, phased genetic maps, and single-strand seq, can provide global phase information to connect local phase blocks to form a chromosome-level phase block. However, it is normal that some small phase blocks cannot be integrated into the large phase block (https://www.nature.com/articles/s41587-020-0719-5) if the global phase data doesn't share any SNPs with the small block. For example, if there is no Hi-C read mapped to a small phase block, then this block cannot be integrated into the large phase block.

For our case, we developed our own workflow for read-based phasing with ONT data, but the workflow by definition should be able to work with other long reads, too:



The idea of our workflow is, first use hapcut2 to build global phase blocks with Hi-C data based on the unphased chromosome-scale assembly, then incorporate the global phase information into whatshap with long reads to produce more dense large phase blocks. Ideally, there should be only one chromosome-level phase block for one chromosome. Whatshap will bin reads to different haplotype groups based on the chromosome-level phase block and then an assembler of choice, for example, flye can be used to quickly assemble each group of reads. After assembly, chimeric contigs are manually corrected by tritex pipeline with the help of Hi-C data, and finally pseudo-chromosomes are constructed by aligning corrected contigs to the unphased reference using a reference-guided scaffolding method, ragtag.

Now, step by step to do read-based phasing:

First step, building phase blocks with Hi-C using hapcut2

I made a new conda environment for hapcut2:

conda create -n hapcut2

Install hapcut2 by conda:

conda install -c bioconda hapcut2 # actually, this installs dependencies required by hapcut2. Hapcut2 itself is a pipeline with the phase algorithms (which are developed by the hapcut2 team)

Also clone the github repository:

git clone https://github.com/vibansal/HapCUT2.git

Hapcut2 team provides a snakemake pipeline to run all the analyses required, so we just need to modify the input accordingly in the snakemake file, here's my snakemake file:

Snakefile

To run snakemake pipeline, do this:

snakemake -s Snakefile --verbose -p

There are many output files, but what we need is the phased vcf file for every chromosome. Then we remove unphased snps from each vcf and finally concatenate vcf files together

```
#!/bin/bash

# extract phased SNPs and combine all the phased SNPs

ls *.VCF > VCF.list

# extract phased SNPs
cat VCF.list | parallel -j 7 -k "bcftools view -p {} > {}.phased.vcf"

# concatenate phased SNPs
bcftools concat *.phased.vcf > Rabiosa_hapcut2_phased_chr.vcf

# compress vcf
bgzip -c Rabiosa_hapcut2_phased_chr.vcf > Rabiosa_hapcut2_phased_chr.vcf.gz
bcftools index Rabiosa_hapcut2_phased_chr.vcf.gz

# reheader vcf, change sample name
bcftools reheader -s sample.name Rabiosa_hapcut2_phased_chr.vcf.gz > Rabiosa_hapcut2_phased_chr_renamed.vcf.gz
```

The content of sample.name is Rabiosa. The aim of renaming sample name in the vcf file is to make all vcf files incorporated in whatshap have the same sample name so whatshap can recognize the same sample in different file. This is important! If whatshap doesn't work for you, check the sample name in vcf file!

Before start whatshap, another thing we need to do is find SNPs in the reference. As it is more reliable to call SNPs with accurate short pair-end reads than error-prone ONT long reads, we use a BWA mem + bcftools approach to call SNPs with pair-end reads:

```
#!/bin/bash

# align PE700 reads to chromosome-level assembly

myread="/home/yutachen/public/Yutangchen/Rabiosa_data/tmp"
fasta="211209_Rabiosa_new_pseudomolecules_v1+unanchored_contigs.fasta"

# index the assembly
bwa index -a bwtsw reference/${fasta}

# align PE700 reads of rabiosa to the assembly
bwa mem -t 48 reference/${fasta} $myread/Ryegrass_SG700bp_trimmed_1P.fq $myread/Ryegrass_SG700bp_trimmed_2P.fq > whatshap/Rabiosa_SG700.sam

# convert sam to bam and sort it
samtools view -Sbh -@ 48 whatshap/Rabiosa_SG700.sam | samtools sort -@ 48 -o whatshap/Rabiosa_SG700_sorted.bam

# call variants
bcftools mpileup -f reference/${fasta} whatshap/Rabiosa_SG700_sorted.bam | bcftools call -mv -Ov -o whatshap/Rabiosa_SG700.vcf

# filter variants, quality > 20 and only bi-allelic SNPs
bcftools filter -i 'QUAL>20' whatshap/Rabiosa_SG700.vcf | bcftools view -g het -v snps -m2 -M2 > whatshap/Rabiosa_SG700_f.vcf

# extract vcf for all chr
bcftools view -t chr1,chr2,chr3,chr4,chr5,chr6,chr7 whatshap/Rabiosa_SG700_f.vcf > whatshap/Rabiosa_SG700_f_chr.vcf

# compress vcf and give new sample name
bgzip -c whatshap/Rabiosa_SG700_f_chr.vcf > whatshap/Rabiosa_SG700_f_chr.vcf.gz
bcftools index whatshap/Rabiosa_SG700_f_chr.vcf.gz

# reheader
bcftools reheader -s whatshap/sample.name whatshap/Rabiosa_SG700_f_chr.vcf.gz > whatshap/Rabiosa_SG700_f_chr_renamed.vcf.gz
```

Of course, in addition to variant detection, we also need to map our ONT long reads to the reference. Instead of using minimap2, the most popular long-read mapping tool, we chose winnowmap, as it is said to map better, https://github.com/marbl/Winnowmap

```
#!/bin/bash

# using winnowmap to align corrected ONT long reads to the phseduo-chromosomes

meryl count k=15 output merylDB ../reference/211209_Rabiosa_new_pseudomolecules_v1+unanchored_contigs.fasta
meryl print greater-than distinct=0.9998 merylDB > repetitive_k15.txt

winnowmap -W repetitive_k15.txt -t 45 -R "@RG\tID:1\tSM:Rabiosa" -ax map-ont ../reference/211209_Rabiosa_new_pseudomolecules_v1+unanchored_contigs.fasta ../../Rabiosa_data/input/ONT_G4_q7a2kf.fq.gz --secondary=no > output.sam

samtools sort -m 3G -@ 30 -o output_sorted.bam output.sam

samtools view -hb -F 2048 -@ 40 -o output_sorted_nosa.bam output_sorted.bam

# extract chr
samtools view -hb -o output_sorted_nosa_chr.bam output_sorted_nosa.bam chr1 chr2 chr3 chr4 chr5 chr6 chr7
```

Notice, when mapping long reads using winnowmap, we add read group name as Rabiosa, just to match the sample name in previous vcf files

Now, start whatshap:

```
#!/bin/bash

# read-based phasing with ONT long reads mapped by winnowmap

whatshap=/home/yutachen/.local/bin/whatshap

# index alignment file
samtools index winnowmap/output_sorted_nosa_chr.bam

# read-based phasing, long reads + hapcut2 global phase information
whatshap phase --sample Rabiosa -o winnowmap/phased.vcf --reference=reference/211209_Rabiosa_new_pseudomolecules_v1.fasta whatshap/Rabiosa_SG700_f_chr_renamed.vcf.gz \
            winnowmap/output_sorted_nosa_chr.bam whatshap/Rabiosa_hapcut2_phased_chr_renamed.vcf.gz

# get some statistics from phase blocks
whatshap stats --gtf winnowmap/phased.gtf --block-list winnowmap/phased.list --tsv winnowmap/phased_stats.tsv winnowmap/phased.vcf

# need to index the phased vcf
bgzip -c winnowmap/phased.vcf > winnowmap/phased.vcf.gz && tabix winnowmap/phased.vcf.gz

# tag reads based the phased SNPs
whatshap haplotag --reference=reference/211209_Rabiosa_new_pseudomolecules_v1.fasta --sample Rabiosa -o winnowmap/tagged.bam \
            --output-haplotag-list winnowmap/tagged_reads.list winnowmap/phased.vcf.gz winnowmap/output_sorted_nosa_chr.bam

# index the tagged bam
samtools index winnowmap/tagged.bam

# extract reads based on read name
# first get names from tagged_reads.list

grep "H1" winnowmap/tagged_reads.list | cut -f1 > winnowmap/H1.list
grep "H2" winnowmap/tagged_reads.list | cut -f1 > winnowmap/H2.list
grep "none" winnowmap/tagged_reads.list | cut -f1 > winnowmap/untagged.list

# extract all the read name
zcat ../Rabiosa_data/input/ONT_G4_q7a2kf.fq.gz | awk 'NR%4==1 {print}' | cut -f1 -d" " | sed 's/@//' > winnowmap/all_read.list

# find unmapped reads (unmapped to chrs)
cd winnowmap
./Find_unmapped_read.py

# use seqtk to extract reads
ls winnowmap/H1.list winnowmap/H2.list winnowmap/untagged.list winnowmap/unmapped.list > myreadlist
cat myreadlist | parallel -j 4 -k "seqtk subseq ../Rabiosa_data/input/ONT_G4_q7a2kf.fq.gz {} > winnowmap/{/.}.fastq"
rm myreadlist
```

What we did through above script is, first, build large and dense phase blocks with phase information provided by long reads and Hi-C, then tag long reads to different haplotypes based on SNPs they span, finally extract reads of different haplotypes based on read names

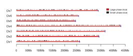
Phasing done! Tutorial of assembly with Flye can be seen from previous blog, Assemble ONT long reads using assembler Flye. Tritex and ragtag deserve one independent blog for each, I will do that later.

Now, it's time to show some results

An example of read tagging from IGV, a region on Rabiosa chr6:

In above plot, blue reads represent one haplotype, red reads represent the other haplotype, and gray reads are those cannot be tagged. It is very interesting to see that blue reads have a very large deletion (the very long black line) on the right side, which doesn't exist in the other haplotype, suggesting phasing or read tagging is correct. We can also zoom in to see more details for this region, for example below plot, we could see SNPs in blue reads have the same allele, and there are two small deletions in blue reads. Reads with the same alleles are tagged as the same haplotype, also suggesting phasing is correct!

Now some statistics of phase blocks and tagged reads:



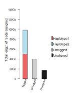
In above plot, the red line means the largest phase block on the chromosome and black lines are small phase block which are not integrated into the large block, and the blue vertical line indicates the total length of the chromosome. So, from this plot, we could see, by combining the global phase provided by Hi-C with local phase provided by long reads, every chromosome ended up with a chromosome-level phase block. However, not surprisingly, there are still small blocks

sample	chromosome	variants	phased	unphased	blocks	variant_per_block_max	bp_per_block_max	bp_per_block_sum	heterozygous_variants	heterozygous_snvs	phased_snvs
Rabiosa	chr1	1064320	1056331	7989	776	1050411	262400071	264624494	1064320	1064320	1056331
Rabiosa	chr2	1333495	1322687	10805	1042	1314780	341736942	345633772	1333495	1333495	1322687
Rabiosa	chr3	1436084	1424888	11193	1010	1416897	367334000	371026499	1436084	1436084	1424888
Rabiosa	chr4	1693703	1680821	12882	1380	1668435	432989167	442563544	1693703	1693703	1680821
Rabiosa	chr5	1065333	1057183	8148	748	1052284	244874204	246243529	1065333	1065333	1057183
Rabiosa	chr6	1092690	1084431	8258	783	1079145	257629914	259232995	1092690	1092690	1084431
Rabiosa	chr7	1359094	1347706	11388	919	1341820	339096701	341651603	1359094	1359094	1347706

Check how many reads are tagged, untagged and unmapped to chromosomes. the *.list files are produced by the whatshap script above.

seqkit stats H1.fastq H2.fastq untagged.fastq unmapped.fastq > tag_reads.stats

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
H1.fastq	FASTQ	DNA	3,703,084	49,511,773,743	2,000	13,370.4	381,245
H2.fastq	FASTQ	DNA	3,571,897	47,741,308,009	2,000	13,365.8	439,975
untagged.fastq	FASTQ	DNA	6,398,652	39,766,446,651	2,000	6,214.8	364,493
unmapped.fastq	FASTQ	DNA	2,003,066	16,474,467,909	2,000	8,224.6	335,711

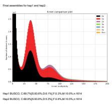


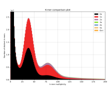
From above plot and table, we could see how much data were assigned to each read group. There are around 49 Gb reads are tagged as haplotype 1 group, which is around 25x haploid genome coverage and 47 Gb reads are assigned to haplotype 2 group, which is also around 25x haploid genome coverage. There are 39 Gb reads untagged and 16 Gb reads are not mapped to chromosomes.

With the haplotype 1 and haplotype 2 reads, I made two assemblies using Flye and raw assembly statistics are in below table. It looks like both assemblies are very fragmented and not complete, I think this is due to the relatively low coverage for each haplotype. So, based on this result, if we want to phase a genome using error-prone ONT data, then we need more data. I guess, we need 200x coverage data.

Assembly	Total length	Number of contigs	N50	N50n
FLYE_H1_misp_10k	1.77 Gb	11384	275 kb	1807
FLYE_H2_misp_10k	1.73 Gb	11375	265 kb	1808

Final assemblies for hap1 and hap2:





Hap1 BUSCO, C:89.7%[S:83.6%,D:6.1%],F:0.3%,M:10.0%,n:1614

Hap2 BUSCO, C:88.9%[S:83.6%,D:5.3%],F:0.6%,M:10.5%,n:1614