Workflows

WGS/WES Mapping to Variant Calls - Version 1.0

The standard workflow for working with DNA sequence data consists of three major steps:

- Mapping
- Improvement
- Variant Calling

Mapping

For reads from 70bp up to a few megabases we recommend using BWA MEM (http://bio-bwa.sourceforge.net/) to map the data to a given reference genome. The reference you use will differ depending on the species your data came from and the resources you want to use with it. For example for a new research project consisting of Human data you would probably use the Genome Reference Consortium's build 38 analysis set (ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Eukaryotes/vertebrates_mammals/Homo_sapiens/GRCh38/seqs_for_alignment_pipelines). Note that with BWA 0.7.10, mapping to alternative haplotypes has been deemed unready for production use, so you will probably wish to use the analysis set that does not contain them.

To prepare the reference for mapping you must first index it by typing the following command where `<ref.fa>` is the path to your reference file:

```
bwa index <ref.fa>
```

This may take several hours as it prepares the Burrows Wheeler Transform index for the reference, allowing the aligner to locate where your reads map within that reference.

Once you have finished preparing your indexed reference you can map your reads to the reference:

```
bwa mem -R '@RG\tID:foo\tSM:bar\tLB:library1' <ref.fa> <read1.fa> <read1.fa> > lane.sam
```

Typically your reads will be supplied to you in two files written in the FASTQ format. It is particularly important to ensure that the @RG information here is correct as this information is used by later tools. The SM field must be set to the name of the sample being processed, and LB field to the library. The resulting mapped reads will be delivered to you in a mapping format known as SAM (http://samtools.github.io/hts-specs/).

Because BWA can sometimes leave unusual FLAG information on SAM records, it is helpful when working with many tools to first clean up read pairing information and flags:

```
samtools fixmate -O bam <lane.sam> <lane_fixmate.bam>
```

To sort them from name order into coordinate order:

```
samtools sort -O bam -o <lane_sorted.bam> -T </tmp/lane_temp> <lane_fixmate.sam>
```

Improvement

In order to reduce the number of miscalls of INDELs in your data it is helpful to realign your raw gapped alignment with the Broad's GATK (https://www.broadinstitute.org/gatk/) Realigner.

```
java -Xmx2g -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -R <ref.fa> -I <lane.bam> -o <lane.intervals> --known <bundle/b38/Mills1000G.b38.vcf
>
java -Xmx4g -jar GenomeAnalysisTK.jar -T IndelRealigner -R <ref.fa> -I <lane.bam> -targetIntervals <lane.intervals> --known <bundle/b38/Mills1000G.b
38.vcf> -o <lane_realigned.bam>
```

BQSR from the Broad's GATK allows you to reduce the effects of analysis artefacts produced by your sequencing machines. It does this in two steps, the first analyses your data to detect covariates and the second compensates for those covariates by adjusting quality scores.

```
java -Xmx4g -jar GenomeAnalysisTK.jar -T BaseRecalibrator -R <ref.fa> -knownSites >bundle/b38/dbsnp_142.b38.vcf> -I <lane.bam> -o <lane_recal.table>
java -Xmx2g -jar GenomeAnalysisTK.jar -T PrintReads -R <ref.fa> -I <lane.bam> --BSQR <lane_recal.table> -o <lane_recal.bam>
```

It is helpful at this point to compile all of the reads from each library together into one BAM, which can be done at the same time as marking PCR and optical duplicates. To identify duplicates we currently recommend the use of either the Picard (http://picard.sourceforge.net/command-line-overview.shtml#MarkDuplicates) or biobambam's (https://github.com/gt1/biobambam) mark duplicates tool.

```
java -Xmx2g -jar MarkDuplicates.jar VALIDATION_STRINGENCY=LENIENT INPUT=<lane_1.bam> INPUT=<lane_2.bam> INPUT=<lane_3.bam> OUTPUT=<library.bam>
```

Once this is done you can perform another merge step to produce your sample BAM files.

```
samtools merge <sample.bam> <library1.bam> <library2.bam> <library3.bam>
samtools index <sample.bam>
```

If you have the computational time and resources available it is helpful to realign your INDELS again:

```
java -Xmx2g -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -R <ref.fa> -I <sample.bam> -o <sample.intervals> --known >bundle/b38/Mills1000G.b3
8.vcf>
java -Xmx4g -jar GenomeAnalysisTK.jar -T IndelRealigner -R <ref.fa> -I <sample.bam> -targetIntervals <sample.intervals> --known >bundle/b38/Mills100
0G.b38.vcf> -o <sample_realigned.bam>
```

Lastly we index our BAM using samtools:

```
samtools index <sample_realigned.bam>
```

Variant Calling

To convert your BAM file into genomic positions we first use mpileup to produce a BCF file that contains all of the locations in the genome. We use this information to call genotypes and reduce our list of sites to those found to be variant by passing this file into bcftools call.

You can do this using a pipe as shown here:

```
samtools mpileup -ugf <ref.fa> <sample1.bam> <sample2.bam> <sample3.bam> | bcftools call -vmO z -o <study.vcf.gz>
```

Alternatively if you need to see why a specific site was not called by examining the BCF, or wish to spread the load slightly you can break it down into two steps as follows:

```
samtools mpileup -go <study.bcf> -f <ref.fa> <sample1.bam> <sample2.bam> <sample3.bam>
bcftools call -vmO z -o <study.vcf.gz> <study.bcf>
```

To prepare our VCF for querying we next index it using tabix:

```
tabix -p vcf <study.vcf.gz>
```

Additionally you may find it helpful to prepare graphs and statistics to assist you in filtering your variants:

```
bcftools stats -F <ref.fa> -s - <study.vcf.gz> > <study.vcf.gz.stats>
mkdir plots
plot-vcfstats -p plots/ <study.vcf.gz.stats>
```

Finally you will probably need to filter your data using commands such as:

```
bcftools filter -O z -o <study_filtered..vcf.gz> -s LOWQUAL -i'%QUAL>10' <study.vcf.gz>
```

Variant filtration is a subject worthy of an article in itself and the exact filters you will need to use will depend on the purpose of your study and quality and depth of the data used to call the variants.

References

- The 1000 Genomes Project Consortium - An Integrated map of genetic variation from 1092 human genomes Nature 491, 56–65 (01 November 2012) doi:10.1038/nature11632 (http://dx.doi.org/10.1038/nature11632)
- GATK Best Practices (http://www.broadinstitute.org/gatk/guide/best-practices)

Using CRAM within Samtools

CRAM is primarily a reference-based compressed format, meaning that only differences between the stored sequences and the reference are stored.

For a workflow this has a few fundamental effects:

1. Alignments should be kept in chromosome/position sort order.
2. The reference must be available at all times. Losing it may be equivalent to losing all your read sequences.

Technically CRAM can work with other orders but it can become inefficient due to a large amount of random access across the reference genome. The current implementation of CRAM in htslib 1.0 is also inefficient in size for unsorted data, although this will be rectified in upcoming releases.

In CRAM format the reference sequence is linked to by the md5sum (M5 auxiliary tag) in the CRAM header (@SQ tags). This is mandatory and part of the CRAM specification. In SAM/BAM format, these M5 tags are optional. Therefore converting from SAM/BAM to CRAM requires some additional overhead to link the CRAM to the correct reference sequence.

A Worked Example

Obtain some public data

We will use the first 100,000 read-pairs from a yeast data set.

```
curl ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR507/SRR507778/SRR507778_1.fastq.gz | gzip -d | head -100000 > y1.fastq
curl ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR507/SRR507778/SRR507778_2.fastq.gz | gzip -d | head -100000 > y2.fastq
curl ftp://ftp.ensembl.org/pub/current_fasta/saccharomyces_cerevisiae/dna/Saccharomyces_cerevisiae.R64-1-1.dna_sm.toplevel.fa.gz | gzip -d > yeast.f
asta
```

Prepare the BWA indices

We need to ensure there exists a .fai fasta index and also indices for whichever aligner we are using (Bwa-mem in this example).

```
samtools faidx yeast.fasta
bwa index yeast.fasta
```

Produce the alignments

The aligner is likely to output SAM in the same order or similar order to the input fastq files. It won't be outputting in chromosome position order, so the output is typically not well suited to CRAM.

```
bwa mem -R '@RG\tID:foo\tSM:bar\tLB:library1' yeast.fasta y1.fastq y2.fastq > yeast.sam
```

The -R option adds a read-group line and applies that read-group to all aligned sequence records. It is not necessary, but a recommended practice.

Sort into chromosome/positon order

Ideally at this point we would be outputting CRAM directly, but at present samtools 1.0 does not have a way to indicate the reference on the command line. We can output to BAM instead and convert (below), or modify the SAM @SQ header to include MD5 sums in the M5: field.

```
samtools sort -O bam -T /tmp -l 0 -o yeast.bam yeast.sam
```

The "-l 0" indicates to use no compression in the BAM file, as it is transitory and will be replaced by CRAM soon. We may wish to use -l 1 if disk space is short and we wish to reduce temporary file size.

Convert to CRAM format

```
samtools view -T yeast.fasta -C -o yeast.cram yeast.bam
```

Note that since the BAM file did not have M5 tags for the reference sequences, they are computed by Samtools and added to the CRAM. In a production environment, this step can be avoided by ensuring that the M5 tags are already in the SAM/BAM header.

The last 3 steps can be combined into a pipeline to reduce disk I/O:

```
bwa mem yeast.fasta y1.fastq y2.fastq | \
samtools sort -O bam -l 0 -T /tmp - | \
samtools view -T yeast.fasta -C -o yeast.cram -
```

Viewing in alignment and pileup format

See the variant calling workflow for more advanced examples.

```
samtools view yeast.cram
samtools mpileup -f yeast.fasta yeast.cram
```

The REF_PATH and REF_CACHE

One of the key concepts in CRAM is that it is uses reference based compression. This means that Samtools needs the reference genome sequence in order to decode a CRAM file. Samtools uses the MD5 sum of the each reference sequence as the key to link a CRAM file to the reference genome used to generate it. By default Samtools checks the reference MD5 sums (@SQ "M5" auxiliary tag) in the directory pointed to by $REF_PATH environment variable (if it exists), falling back to querying the European Bioinformatics Institute (EBI) reference genome server, and further falling back to the @SQ "UR" field if these are not found.

While the EBI have an MD5 reference server for downloading reference sequences over http, we recommend use of a local MD5 cache. We have provided with Samtools a basic script (misc/seq_cache_populate.pl) to convert your local yeast.fasta to a directory tree of reference sequence MD5 sums:

```
<samtools_src_dir>/misc/seq_cache_populate.pl -root /some_dir/cache yeast.fasta
export REF_PATH=/some_dir/cache/%2s/%2s/%s:http://www.ebi.ac.uk/ena/cram/md5/%s
export REF_CACHE=/some_dir/cache/%2s/%2s/%s
```

REF_PATH is a colon separated list of directories in which to search for files named after the sequence M5 field. The : in http:// is not considered to be a separator. Hence using the above setting, any CRAM files that are not cached locally may still be looked up remotely.

In this example "%2s/%2s/%s" means the first two digits of the M5 field followed by slash, the next two digits and slash, and then the remaining 28 digits. This helps to avoid one large directory with thousands of files in it.

The REF_CACHE environment variable is used to indicate that any downloaded reference sequences should be stored locally in this directory in order to avoid subsequent downloads. This should normally be set to the same location as the first directory in REF_PATH.

Copyright © 2017 Genome Research Limited (reg no. 2742969) is a charity registered in England with number 1021457. Terms and conditions (/terms).