SAMtools

Calling SNPs/INDELs with SAMtools/BCFtools

The basic Command line

Suppose we have reference sequences in *ref.fa*, indexed by **samtools faidx**, and position sorted alignment files *aln1.bam* and *aln2.bam*, the following command lines call SNPs and short INDELs:

```
samtools mpileup -ugf ref.fa aln1.bam aln2.bam | bcftools view -bvcg - > var.raw.bcf
bcftools view var.raw.bcf | vcfutils.pl varFilter -D100 > var.flt.vcf
```

where the **-D** option sets the maximum read depth to call a SNP. SAMtools acquires sample information from the **SM** tag in the **@RG** header lines. One alignment file can contain multiple samples; reads from one sample can also be distributed in different alignment files. SAMtools will regroup the reads anyway. In addition, if no **@RG** lines are present, each alignment file is taken as one sample.

Understanding the command line

In the command line above, **samtools** collects summary information in the input BAMs, computes the likelihood of data given each possible genotype and stores the likelihoods in the BCF format (see below). It does not call variants.

Bcftools applies the prior and does the actual calling. It can also concatenate BCF files, index BCFs for fast random access and convert BCF to VCF. In addition, **bcftools** can operate on some VCFs (e.g. calling SNPs from GL-tagged VCFs), but not for all VCFs; VCF to BCF conversion is not working at the moment, either.

Tuning the parameters

One should consider to apply the following parameters to **mpileup** in different scenarios:

- Apply **-C50** to reduce the effect of reads with excessive mismatches. This aims to fix overestimated mapping quality and appears to be preferred for BWA-short.

- Given multiple technologies, apply **-P** to specify which technologies to use for collecting initial INDEL candidates. It is recommended to find INDEL candidates from technologies with low INDEL error rate, such as Illumina. When this option is in use, the value(s) following the option must appear in the **PL** tag in the **@RG** header lines.
- Apply **-D** and **-S** to keep per-sample read depth and strand bias. This is preferred if there are more than one samples at high coverage.
- Adjust **-m** and **-F** to control when to initiate indel realignment (requiring r877+). Samtools only finds INDELs where there are sufficient reads containing the INDEL at the same position. It does this to avoid excessive realignment that is computationally demanding. The default works well for many low-coverage samples but not for, say, 500 exomes. In the latter case, using **-m 3 -F 0.0002** (3 supporting reads at minimum 0.02% frequency) is necessary to find singletons.
- Use `**-BQ0 -d10000000 -f ref.fa**' if the purpose is to get the precise depth of coverage rather than call SNPs. Under this setting, **mpileup** will count low-quality bases, process all reads (by default the depth is capped at 8000), and skip the time-demanding BAQ calculation.
- Apply **-A** to use anomalous read pairs in **mpileup**, which are not used by default (requring r874+).

Understanding the output: the VCF/BCF format

The VCF format

The Variant Call Format (VCF) is the emerging standard for storing variant data. Originally designed for SNPs and short INDELs, it also works for structural variations.

VCF consists of a header section and a data section. The header must contain a line starting with one '#', showing the name of each field, and then the sample names starting at the 10th column. The data section is TAB delimited with each line consisting of at least 8 mandatory fields (the first 8 fields in the table below). The FORMAT field and sample information are allowed to be absent. We refer to the official VCF spec for a more rigorous description of the format.

Col	Field	Description
1	CHROM	Chromosome name
2	POS	1-based position. For an indel, this is the position preceding the indel.
3	ID	Variant identifier. Usually the dbSNP rsID.
4	REF	Reference sequence at POS involved in the variant. For a SNP, it is a single base.

5	ALT	Comma delimited list of alternative seuqence(s).
6	QUAL	Phred-scaled probability of all samples being homozygous reference.
7	FILTER	Semicolon delimited list of filters that the variant fails to pass.
8	INFO	Semicolon delimited list of variant information.
9	FORMAT	Colon delimited list of the format of individual genotypes in the following fields.
10+	Sample(s)	Individual genotype information defined by FORMAT.

The BCF Format

BCF, or the binary variant call format, is the binary version of VCF. It keeps the same information in VCF, while much more efficient to process especially for many samples. The relationship between BCF and VCF is similar to that between BAM and SAM. The detailed format description of the BCF format can be found in *bcf.tex* included in the samtools source code package.

SAMtools/BCFtools specific information

SAMtools/BCFtools may write the following tags in the *INFO* field in VCF/BCF.

Tag	Description			
I16	16 integers:			
	1	#reference Q13 bases on the forward strand	2	#reference Q13 bases on the reverse strand
	3	#non-ref Q13 bases on the forward strand	4	#non-ref Q13 bases on the reverse strand
	5	sum of reference base qualities	6	sum of squares of reference base qualities
	7	sum of non-ref base qualities	8	sum of squares of non-ref base qualities
	9	sum of ref mapping qualities	10	sum of squares of ref mapping qualities
	11	sum of non-ref mapping qualities	12	sum of squares of non-ref mapping qualities
	13	sum of tail distance for ref bases	14	sum of squares of tail distance for ref bases
	15	sum of tail distance for non-ref bases	16	sum of squares of tail distance for non-ref

INDEL	Indicating the variant is an INDEL.
DP	The number of reads covering or bridging POS.
DP4	Number of 1) forward ref alleles; 2) reverse ref; 3) forward non-ref; 4) reverse non-ref alleles, used in variant calling. Sum can be smaller than DP because low-quality bases are not counted.
PV4	P-values for 1) strand bias (exact test); 2) baseQ bias (t-test); 3) mapQ bias (t); 4) tail distance bias (t)
AF1	EM estimate of the site allele frequency of the strongest non-reference allele.
CI95	Equal-tail (Bayesian) credible interval of the site allele frequency at the 95% level.
HWE	P-value for Hardy-Weinberg equillibrium (chi-square test)

- For a read, suppose the k-th base is overlapping the reference base we are looking at. The tail distance for this read equals min{k-1,l-k} where l is the length of the read. If tail distance is small, the difference on the read may be caused by undetected indels.
- In the BCF generated by SAMtools, an non-ref base 'X' represents an base has not been seen from the alignment data. Such a base is necessary to evaluate the probability of missing a non-ref allele due to sampling fluctuation.
- SAMtools/BCFtools writes genotype likelihoods in the PL format which is a comma delimited list of phred-scaled data likelihoods of each possible genotype. For example, suppose REF=C and ALT=A,G, PL=7,0,13,37,40,49 means for the sample we are looking at, P(D|CC)=10^{-0.7}, P(D|CA)=1, P(D|CG)=10^{-1.3}, P(D|AA)=10^{-3.7}, P(D|AG)=1e-4 and P(D|GG)=10^{-4.9}.

Estimating the Allele Frequency Spectrum

In addition to SNP/INDEL calling, **bcftools** is also able to estimate the Allele Frequency Spectrum (AFS) using the same model for variant calling.

While calling SNPs, **bcftools** will print the estimated AFS to the error output at lines starting with `[afs]`. However, to accurately estimate AFS, we need to iterate the procedure. In most applications, we are only interested in the AFS conditional on a list of loci. Suppose we have the list of loci in file *cond.txt* with the first two columns in the file giving the coordinates, the procedure to estimate AFS is:

```
bcftools view -NIbl cond.txt data.bcf > cond.bcf
bcftools view -cGP cond2 cond.bcf > round1.vcf 2> round1.afs
bcftools view -cGP round1.afs cond.bcf > /dev/null 2> round2.afs
bcftools view -cGP round2.afs cond.bcf > /dev/null 2> round3.afs
......
```

until the AFS converges, which usually takes less than 10 rounds of EM iterations. The first command line above extracts sites in *cond.txt* for efficiency in later steps. Option **-P** specifies the initial AFS (in SNP calling, this is prior), which can be a file (as in the 3rd and 4th command lines) or '*full*', '*cond2*' or '*flat*' (as in the 2nd command line). Choosing the right initial AFS helps accuracy and reduces iterations and potential overfitting.

Another way to estimate AFS is to get the list of site allele frequency by

```
perl -ne 'print "$1\n" if /AF1=([^,;]+)/' round1.vcf
```

and then derive the histogram. It should be noted that AF1 does not use the initial AFS. No iterations are needed. However, the histogram of AF1 usually has a lot of noises. It is up to the users to decide which method to use.

Base Alignment Quality (BAQ)

Base Alignment Quality (BAQ) is a new concept deployed in samtools-0.1.9+. It aims to provide an efficient and effective way to rule out false SNPs caused by nearby INDELs. The following shows the alignments of 6 reads by a typical read mapper in the presence of a 4bp homozygous INDEL:

```
coor     12345678901234    5678901234567890123456
ref      aggttttataaaac----aattaagtctacagagcaacta
sample   aggttttataaaacAAATaattaagtctacagagcaacta
read1    aggttttataaaac****aaAtaa
read2     ggttttataaaac****aaAtaaTt
read3         ttataaaacAAATaattaagtctaca
read4             CaaaT****aattaagtctacagagcaac
read5               aaT****aattaagtctacagagcaact
read6                 T****aattaagtctacagagcaacta
```

where capital bases represent differences from the reference and underlined bases are the inserted bases. The alignments except for read3 are wrong because the 4bp insertion is misplaced. The mapper produces such alignments because when doing a pairwise alignment, the mapper prefers one or two mismatches over a 4bp insertion. What is hurting more is that the wrong alignments lead to recurrent mismatches, which are likely to deceive most site-independent SNP callers into calling false SNPs.

One way to avoid such an artifact is to do multi-sequence realignment, but the current implementations are very computationally demanding. SAMtools seeks another solution. It assigns each base a BAQ which is the Phred-scaled probability of the base being misaligned. BAQ is low if the base is aligned to a different reference base in a suboptimal alignment, and in this case a mismatch should contribute little to SNP calling even if the base quality is high. With BAQ, the mismatches in the example above are significantly downweighted. SAMtools will not call SNPs from that.

The BAQ strategy is invoked by default in **mpileup**. To make other SNP callers take advantage of BAQ, one should run

```
samtools calmd -Abr aln.bam ref.fa > aln.baq.bam
```

to cap base quality by BAQ and then give *aln.baq.bam* to the SNP callers as the input. For high-coverage single-sample SNP calling, BAQ appears to be as effective as multi-sequence realignment, while being much faster and easier to use. Currently the BAQ strategy is the only practical way to avoid the INDEL artifact in low-coverage multi-sample SNP calling.

Limitations

- BCFtools does not properly handle multi-allelic variants. It only takes the strongest non-reference allele.
- SAMtools does not properly compute the likelihoods of multi-allelic INDELs.
- The VCF file produced by BCFtools does not strictly conform the VCF spec. For example, the GT genotype information is not always present because for the purpose of BCF, GT is unnecessary and takes disk space. In addition, GT is not the first as is required by the VCF spec. This can be fixed by the **bcf-fix.pl** script that comes with the source code package, and will be fixed in future (fixed in r880+).

Appendix: Use Cases

SNP/INDEL calling for hundreds of exomes

The following shows the detailed procedure on how to call SNPs/INDELs for hundreds of exomes. It only aims to provide an overview of how to handle huge data sets. Some command lines given below may not work for all systems. Advanced users may also want to modify based on their own system

configurations.

In the following, the key and the most difficult part is the command line calling **samtools mpileup**. Once that is done, one can use 3rd party tools or write their own scsripts to achieve the rest.

Input and preparation

- Obtain auxiliary scripts/programs: array job submission script **asub**, basic BED processing utility **bedutils.pl**, alphanumeric sort and **tabix**:

```
export PATH="$HOME/bin:$PATH"
wget http://lh3lh3.users.sourceforge.net/download/asub
wget http://lh3lh3.users.sourceforge.net/download/bedutils.pl
cp asub bedutils.pl $HOME/bin
wget http://lh3lh3.users.sourceforge.net/download/sort-20101217.tar.bz2
tar -jxf sort-20101217.tar.bz2
(cd sort-20101217; make; cp sort $HOME/bin/sort-alt)
wget http://sourceforge.net/projects/samtools/files/tabix/tabix-0.2.3.tar.bz2/download
tar -jxf tabix-0.2.3.tar.bz2;
(cd tabix-0.2.3; make; cp tabix bgzip $HOME/bin)
```

- Check out a more recent version of samtools

```
svn co https://samtools.svn.sourceforge.net/svnroot/samtools/trunk/samtools
(cd samtools; make; cp samtools bcftools/{bcftools,vcfutils.pl} $HOME/bin)
```

- The reference sequence file *ref.fa*. To avoid potential mistakes, *ref.fa* is better the one that is used for read mapping. File *ref.fa* should be indexed by samtools' faidx, if this has not been done.

```
samtools faidx ref.fa
```

- File *bam.list* which gives the paths of indexed BAM files, one line per file. These files should be better tagged with read groups; if not, one BAM per sample.
- File *target.bed* which gives the target regions in the BED format.

Procedure

- Calling SNPs/INDELs in small regions

```
vcfutils.pl splitchr -l 500000 | xargs -i \
  echo samtools mpileup -C50 -m3 -F0.0002 -DSuf ref.fa -r {} -b bam.list \| bcftools \
  view -bcvg - \> part-{}.bcf | aub -j run-part
```

- Merge regions

```
cut -f1 ref.fa.fai | xargs -i echo grep ^part-{} \| sort-alt -N \| xargs \
  bcftools cat \> chr-{}.bcf
awk '{print "chr-"$1".bcf"}' ref.fa.fai | xargs bcftools cat > merge.bcf
bcftools index merge.bcf
```

To make sure *merge.bcf* is correctly generated, one should use bcftools view to retrieve data towards the end of the last chromosome. If something goes wrong in data processing, no data will be retrieved. When we confirm *merge.bcf* is correct, we may delete intermediate files *part-*.bcf* and *chr-*.bcf*.

- Extract target regions

```
bcftools view -G merge.bcf | vcfutils.pl varFilter -w0 -10 -20 -30 -40 -Q0 \
  | bgzip > merge.vcf.gz
tabix -fpvcf merge.vcf.gz
sort-alt -k1,1N -k2,2n target.bed | bedutils.pl union \
  | awk '{print $1":"($2+1)"-"$3}' | xargs tabix merge.vcf.gz \
  | bgzip > target.vcf.gz
bcftools view -bl target.vcf.gz > target.bcf
```

Here we use **tabix** to retrieve target regions, which is a little complicated. In future new functionality may be added to **bcftools** to simplify this step.

- Filtering. For now, an effective way of filtering is unknown. For indels, one may consider the following:

```
bcftools view -GM target.bcf | grep INDEL \
  | vcfutils.pl varFilter -10 -20 -30 -40 -a4 -G90 -S30
```

although more effective filters may be existing. For INDELs, one may use the fraction of frameshift indels as a proxy to specificity.

Last modified: 2010-12-17