TECHNICAL NOTE

DiscoSnp++: Additional File

Pierre Peterlongo*, Erwan Drezen, Claire Lemaitre and Chloé Riou

*Correspondence:
pierre.peterlongo@inria.fr
GenScale, INRIA Rennes Bretagne-Atlantique, IRISA, Campus de Beaulieu, Rennes, France
Full list of author information is available at the end of the article

Validation of predictions and precision/recall computations

Both on real datasets or synthetic ones, predicted polymorphisms can be compared to a reference set. In all the performed tests, one dispose from a reference genome that is used in the purpose of simulating and validating predictions. Whatever the tested method, the predicted polymorphisms are validated using the following process:

- All predicted sequences are fully mapped using Gassst [1] on the reference genome with a least 90% similarity.
- For SNPs:
  - For each couple of sequences predicting one or more SNPs: if one or the two sequence(s) map the reference genome: for each predicted SNP of the mapped sequence(s), if its position matches exactly a position on which a SNP was simulated, then this SNP is considered as a True Positive (TP).
- For indels:
  - If one or the two sequence(s) of a predicted indel matches a simulated indel position, then such a prediction is considered as a True Positive (TP).
- A predicted polymorphism that is not a True Positive is a False Positive (FP).
- A simulated polymorphism not mapped by a predicted polymorphism is a False Negative (FN).

The precision defined by

$$precision = \frac{numberTP}{numberTP + numberFP}$$

and the recall is defined by

$$recall = \frac{numberTP}{numberTP + numberFN}.$$

Note that in the case of real datasets, the *precision* is meaningless as the reference sets are not exhaustive. The main manuscript does not provide *precision* for real read sets.

Simulation experiments

Two to n E. Coli datasets: simulations and experiments

Simulations

We propose an experiment simulating more than two haploid bacterial individuals. For doing this, we created 30 copies (that we call individuals) of the E. coli K-12

MG1655 strain. We then simulated SNPs with a uniform distribution such that ≈ 4200 SNPs (≈0.1% of the genome length) are common to any pair of individuals, half this number is common to any trio of individuals, a third to any quadruplet, and so on. With this strategy, while considering all the 30 individuals together, 69 600 SNP sites were generated, covering ≈1.5% of the genome. We also simulated indels following exactly the same process, with a ratio of one indel for ten SNPs.

We simulated a 40x sequencing of each of the 30 individuals, with 100-bp reads and 0.1% error rate. Thus, 1,855,870 reads were generated per read set.

Experiments

Experiments were made on subsets of two Coli individuals (the two first read sets), three individuals (the three first read sets), and so on up to 30 experiments. The command used are reported Table 1.

Two human read sets: simulations and experiments

Simulations

An experiment was performed on two human diploid read sets. This experiment applied on human chromosome 1, (GRCh37/hg19 reference assembly version), ≈ 249 million base pairs. SNPs and indels were simulated following the 1000 genome project [2] predictions. Precisely, from the phase 1 vcf file [3], we extracted SNPs and indels of individuals HG00096 and HG00100. Variants having the same genotypes in both individuals were discarded. For each of the two individuals, two version of the chromosome were created. On each of these two versions, SNPs and indels were placed following the VCF file, i.e., 0|0: no modification of the chromosomes, 0|1: modification of one of the two chromosomes (randomly chosen), 1|1: modification of the two chromosomes.

For each individual, we simulated a 40x illumina sequencing with 0.1% error rate (20x per chromosome). Thus, 99,600,000 reads were generated per read set.

Experiments

Experiments were performed using *DiscoSnp++*, *cortex* and the hybrid approach using the commands indicated Table 2.

GATK parameters

As mentioned in the manuscript, we used GATK either simply calling SNPs and indel using the *UnifiedGenotyper* option as shown Table 2 or using the reads realignment and the *HaplotypeCaller*.

We decided not to present results using the reads realignment and the *HaplotypeCaller* as this does not improve significantly the prediction precision and recall (see Figure 1) and as the execution time is three times longer (from approximately 19h to 54h for this experiment).

Saccharomyces cerevisiae: experiment

We used reads provided by the Kvitek [4] study. The 24 read sets were downloaded from the NCBI Sequence Read Archive (with the accession number SRA054922), and processed to remove barcode and adapter sequences as in the initial study.

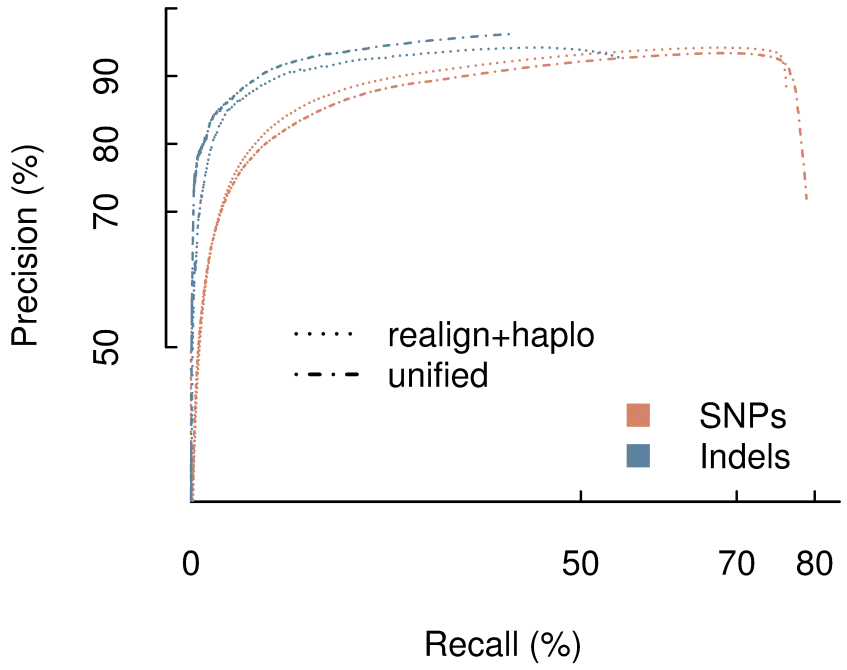

Figure 1 Comparison of the hybrid results on the human dataset depending on the GATK pipeline. "*realign+haplo*" stands for the using the GATK *realigner* and of the *HaplotypeCaller*, while "unified" stands for the *UnifiedGenotyper* only.

DiscoSnp++ was run independently on populations E1, E2 and E3. For each population, *DiscoSnp++* was applied on the eight read sets corresponding to the eight time points, with the default parameters and $c = 11$, $D = 60$ (searching for indels of length at most 60) and $P = 4$ (authorizing up to 4 close SNPs in a unique bubble).

The command were the following:

```
run_discoSnp++.sh -r "`ls data/E1_gen*`" -D 60 -P 4 -c 11
run_discoSnp++.sh -r "`ls data/E2_gen*`" -D 60 -P 4 -c 11
run_discoSnp++.sh -r "`ls data/E3_gen*`" -D 60 -P 4 -c 11
```

1 VCF creation

1.1 Mapping predictions on a reference genome

Depending on the user choice and the availability of a reference genome, *DiscoSnp++* may map its predictions. We performed tests using BWA [5] and we provide an automatic pipeline based on this tool output.

The reference genome and the predictions may present differences due to distinct individuals, bad assembly of the reference or reference species distant from analyzed reads. Thus, errors (indel and mismatches) are authorized during the mapping. By default we use BWA authorizing at most $t = 4$ errors and using a seed of length 10. Each prediction (each couple of sequence $\{S_A,S_B\}$) is considered as mapped at a unique genomic position (and therefor validated) if there exists a minimal distance $d \leq t$ for which S_A and/or S_B have a unique match on the reference genome. In this case, the position of this match is indicated in the VCF file. Algorithm 1, presents how this unique position, if exists, is detected for each predicted variant.

Algorithm 1 Mapping and validation of mapped variants

Map *DiscoSnp++* output on the reference genome, authorizing at most t errors (indels and mismatches)
for each *DiscoSnp++* prediction p (couple of sequence $\{S_A,S_B\}$) **do**
  for each position of the prediction **do**
    if d (mapping distance) $<= D$ (best mapping distance) **then**
      Keep the mapping distance as best mapping distance
    end if
  end for
  for each position of the prediction **do**
    if $d == D$ **then**
      Compute $U = \{$genomic positions where S_A and/or S_B mapped at distance $== D\}$
    end if
  end for
  if $|U| == 1$ **then**
    Set prediction p as Mapped at this unique position (within distance d)
    Break
  end if
  if $|U| > 1$ **then**
    Set prediction p Multiple Mapped
    Break
  end if
end for
if p never set as "Mapped" (Unique position or Multiple Mapping) **then**
  set p as "Unmapped"
end if

1.2 Detailed VCF content

The proposed VCF file contains the following fields:

- **CHROM**: Chromosome id where the prediction is mapped, or '.' if no reference genome provided or if no (unique) mapping of the variant
- **POS**:
  - If a reference genome is provided and if the couple is mapped on a unique position: position of the variant (with the first base having the position 1)
  - If a reference genome is provided and if the couple is not uniquely mapped: one of the positions of the variant (the first reported by BWA)
  - Else (no reference genome provided or couple unmapped):position on the local assembly predict by *DiscoSnp++* (path, unitig or contig)
- **ID**: identification of the SNP or the indel.
- **REF**:

  For SNPs:
  - If one of the two predictions is mapped on the genome: the nucleotide present on the reference genome at the variant position.
  - Else, or if no reference genome provided: the lexicographically smallest of the two variants
  - In case of close SNPs : the first is defined as previously described. The following SNPs are those located on the same path

  For Indels: one of the two predictions is longer than the other.
  - If the smallest prediction is mapped on the genome: the nucleotide present on the reference genome at the variant position-1.
  - If the longest prediction is mapped on to the genome: the sequence of the reference genome from the variant position-1 to the variant position+size of the indel.

  - Else, or if no reference genome provided: the lexicographically smallest of the two variants

  Remark: the REF (and ALT) sequences are extracted from the reference genome is forward strand. If the reverse complement of the prediction is mapped, the reported REF and ALT sequences correspond to the reverse complement of the *DiscoSnp++* predictions.
- **ALT**: The variant non reported as the "REF" variant
- **QUAL**: '.'
- **FILTER**:
  - PASS if the variant is mapped on a unique position
  - MULTIPLE if the variant is mapped on multiple positions
  - '.' if no reference genome provided
- **INFO**:
  - **Ty**: Type of variant: 'SNP', 'INS' or 'DEL'
  - **Rk**: Rank of the prediction computed by *DiscoSnp++*
  - **UL**: Length of the left unitig ('None' if not computed)
  - **UR**: Length of the right unitig ('None' if not computed)
  - **CL**: Length of the left contig ('None' if not computed)
  - **CR**: Length of the right contig ('None' if not computed)
  - **Genome**: Applies only for SNPs when a reference genome is provided ('.' for indels and when no reference genome provided). Nucleotide on the reference genome at the variant position. **Important Remark**: This is equal to the "REF" field only if one of the two prediction matches the reference.
  - **Sd**: Applies only when a reference genome is provided ('.' if no reference genome provided or if the variant was not mapped). Strand of the prediction mapping. '1': Forward, '-1': Reverse. **Important Remark**: Fields "REF", "ALT" and "AR" are based on the mapped predictions.
- **FORMAT**: specificying the data type and the order of this data.
  - **GT**:Genotype for the REF prediction (given by discoSnp++). The allele are separated by '—' in case of snp and indel (genotype unphased) or by '/' for close snps (genotype phased)
  - **DP**:Cumulated depth accross samples (sum)
  - **PL**:Phred-scaled Genotype Likelihoods
  - **AD**:Depth of each allele by sample
- **GENOTYPE**:all informations of the format field by sample

2 Genotyping

$$c_u : \text{Coverage upper path}$$
$$c_l : \text{Coverage lower path}$$
$$err : \text{Uniform error rate}(0.01)$$
$$prior : \text{heterozygous prior}(0.33)$$
$$like_{0/0} = -10\log_{10}\left((1-err)^{c_u} \times err^{c_l} \times C^{c_u}_{c_u+c_l} \times \frac{1-prior}{2}\right)$$
$$like_{1/1} = -10\log_{10}\left(err^{c_u} \times (1-err)^{c_l} \times C^{c_u}_{c_u+c_l} \times \frac{1-prior}{2}\right)$$
$$like_{0/1} = -10\log_{10}\left(\left(\frac{1}{2}\right)^{c_u+c_l} \times prior\right)$$
$$min(like_{0/0}, like_{1/1}, like_{0/1}) \Rightarrow 0/0, \text{or } 1/1, \text{or } 0/1$$

3 Algorithm: Bubble walking

Algorithm 2 SNP & Indels caller

#Search for SNPs
for each couple of successor kmers $kmer_1, kmer_2$ **do**
    expand($kmer_1, kmer_2$, 1 SNPs already detected up to P authorized close SNPs)
end for
#Search for Indels (whatever predicted SNPs)
for each path selected as the $extended\_path$ **do**
    Breadth first extend the $extended\_path$ (up to D nuc.)
    if Last character of the extension == last char. of the non expanded $kmer$ **then**
        expand($kmer_1, kmer_2$, 1 variant already detected up to 1 authorized variant)
    end if
end for

Algorithm 3 Expand two kmers

Test branching $kmer_1, kmer_2$. **Exit if necessary**
Get $kmer_1^+$ and $kmer_2^+$ the unique possible extensions of $kmer_1$ and $kmer_2$ with the same α. Note that with branching=2: there may exist several couple $kmer_1^+$, $kmer_2^+$, both extended with the same α
if $kmer_1^+ == kmer_2^+$ **then**
    Finish the bubble & **stop the expand**
else
    Expand($kmer_1^+$, $kmer_2^+$, 0 new SNP detected up to P authorized close SNPs)
end if
if Number detected close SNPs $< P$ **then**
    for $kmer_1^+$ and $kmer_2^+$ two kmers extending $kmer_1$ and $kmer_2$ with distinct α, β **do**
        if $kmer_1^+ == kmer_2^+$ **then**
            Finish the bubble & **stop the expand**
        else
            Expand($kmer_1^+$, $kmer_2^+$, 1 new SNP detected up to P authorized close SNPs)
        end if
    end for
end if

References

1. Rizk, G., Lavenier, D.: GASSST: global alignment short sequence search tool. Bioinformatics (Oxford, England) **26**(20), 2534–40 (2010). doi:10.1093/bioinformatics/btq485
2. Altshuler, D.M., Durbin, R.M., Abecasis, G.R., Bentley, D.R., Chakravarti, A., Clark, A.G., Donnelly, P., Eichler, E.E., Flicek, P., Gabriel, S.B., Gibbs, R.A., Green, E.D., Hurles, M.E., Knoppers, B.M., Korbel, J.O., Lander, E.S., Lee, C., Lehrach, H., Mardis, E.R., Marth, G.T., Mcvean, G.A., Nickerson, D.A., Schmidt, J.P., Sherry,

S.T., Wang, J., Wilson, R.K., Dinh, H., Kovar, C., Lee, S., Lewis, L., Muzny, D., Reid, J., Wang, M., Fang, X.D., Guo, X.S., Jian, M., Jiang, H., Jin, X., Li, G.Q., Li, J.X., Li, Y.R., Li, Z., Liu, X., Lu, Y., Ma, X.D., Su, Z., Tai, S.S., Tang, M.F., Wang, B., Wang, G.B., Wu, H.L., Wu, R.H., Yin, Y., Zhang, W.W., Zhao, J., Zhao, M.R., Zheng, X.L., Zhou, Y., Gupta, N., Clarke, L., Leinonen, R., Smith, R.E., Zheng-Bradley, X., Grocock, R., Humphray, S., James, T., Kingsbury, Z., Sudbrak, R., Albrecht, M.W., Amstislavskiy, V.S., Borodina, T.A., Lienhard, M., Mertes, F., Sultan, M., Timmermann, B., Yaspo, M.L., Fulton, L., Fulton, R., Weinstock, G.M., Balasubramaniam, S., Burton, J., Danecek, P., Keane, T.M., Kolb-Kokocinski, A., McCarthy, S., Stalker, J., Quail, M., Davies, C.J., Gollub, J., Webster, T., Wong, B., Zhan, Y.P., Auton, A., Yu, F., Bainbridge, M., Challis, D., Evani, U.S., Lu, J., Nagaswamy, U., Sabo, A., Wang, Y., Yu, J., Coin, L.J.M., Fang, L., Li, Q.B., Li, Z.Y., Lin, H.X., Liu, B.H., Luo, R.B., Qin, N., Shao, H.J., Wang, B.Q., Xie, Y.L., Ye, C., Yu, C., Zhang, F., Zheng, H.C., Zhu, H.M., Garrison, E.P., Kural, D., Lee, W.P., Leong, W.F., Ward, A.N., Wu, J.T., Zhang, M.Y., Griffin, L., Hsieh, C.H., Mills, R.E., Shi, X.H., von Grotthuss, M., Zhang, C.S., Daly, M.J., DePristo, M.A., Banks, E., Bhatia, G., Carneiro, M.O., del Angel, G., Genovese, G., Handsaker, R.E., Hartl, C., McCarroll, S.A., Nemesh, J.C., Poplin, R.E., Schaffner, S.F., Shakir, K., Yoon, S.C., Lihm, J., Makarov, V., Jin, H.J., Kim, W., Kim, K.C., Rausch, T., Beal, K., Cunningham, F., Herrero, J., McLaren, W.M., Ritchie, G.R.S., Gottipati, S., Keinan, A., Rodriguez-Flores, J.L., Sabeti, P.C., Grossman, S.R., Tabrizi, S., Tariyal, R., Cooper, D.N., Ball, E.V., Stenson, P.D., Barnes, B., Bauer, M., Cheetham, R.K., Cox, T., Eberle, M., Kahn, S., Murray, L., Peden, J., Shaw, R., Ye, K., Batzer, M.A., Konkel, M.K., Walker, J.A., MacArthur, D.G., Lek, M., Herwig, R., Shriver, M.D., Bustamante, C.D., Byrnes, J.K., De la Vega, F.M., Gravel, S., Kenny, E.E., Kidd, J.M., Lacroute, P., Maples, B.K., Moreno-Estrada, A., Zakharia, F., Halperin, E., Baran, Y., Craig, D.W., Christoforides, A., Homer, N., Izatt, T., Kurdoglu, A.A., Sinari, S.A., Squire, K., Xiao, C.L., Sebat, J., Bafna, V., Burchard, E.G., Hernandez, R.D., Gignoux, C.R., Haussler, D., Katzman, S.J., Kent, W.J., Howie, B., Ruiz-Linares, A., Dermitzakis, E.T.: An integrated map of genetic variation from 1,092 human genomes. Nature **491**, 56–65 (2012). doi:10.1038/nature11632

3. 1000 genome phase 1 vcf file. ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20110521/ALL.chr1.phase1$_{r}elease_{v}3.20101123.snps_{i}ndels_{s}vs.genotypes.vcf.gz$
4. Kvitek, D.J., Sherlock, G.: Whole genome, whole population sequencing reveals that loss of signaling networks is the major adaptive strategy in a constant environment. PLoS genetics **9**(11), 1003972 (2013). doi:10.1371/journal.pgen.1003972
5. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics **25**(14), 1754–1760 (2009). doi:10.1093/bioinformatics/btp324

DiscoSnp++

```
for ((i=2;i<=31;i++)); do
#Prepare input parameters
 list=""; for((j=0;j<i;j++)); do list=$list" "coli_muted_n_30_genome_$j\_reads.fasta; done
#Run discoSnp++
 run_discoSnp++.sh -r  "$list" -p dsc_$i\_genomes -P 4 -D 10
done
```

cortex

```
#(needs a compilation per number of input read sets):
for ((i=1;i<31;i++)); do make MAXK=31 NUM_COLS=$i cortex_var; done

#Prepare input parameters
for ((i=0;i<30;i++)); do echo coli_muted_n_30_genome_$i\_reads.fasta > ind$i; done
for ((i=2;i<31;i++)); do
 for ((j=0;j<i;j++)); do echo -e "IND$j\tind$j\t.\t." >> INDEX$i; done;
done

#Prepare data
for ((i=0;i<30;i++)); do
 cortex_var_31_c2 --se_list ind$i --kmer_size 31 --mem_height 20 --mem_width 75 \
  --dump_binary genome_$i\_reads.ctx  --sample_id sample$i --remove_low_coverage_supernodes 3
done

for ((i=0;i<30;i++)); do
#Prepare input parameters
 echo  "genome_$i\_reads.ctx" > ind$i\_ctx
 for ((i=0;i<$nb;i++));
  do echo ind$i\_ctx >> my_cortex_list_ctx_$nb
 done
#Detect variants
 cortex_var_31_c$nb --colour_list my_cortex_list_ctx_$nb --kmer_size 31 --mem_height 22 --mem_width 75 \
  --detect_bubbles1 0/1 --output_bubbles1 my_res_cortex_$nb --remove_low_coverage_supernodes 3
done
```

Hybrid approach

```
#Assembly
SOAPdenovo-63mer pregraph -s soap.config  -o soapNAR30A -K $k -d 5
SOAPdenovo-63mer contig -g soapNAR30A

#remove contigs whose size is < 100:
python ./filter_fasta_by_length.py soapNAR30A.contig NAR30A_ref.contigs.fa 100
mv  soapNAR30A.contig  NAR30A_ref.contigs.fa
bowtie2-build -f NAR30A_ref.contigs.fa NAR30A_ref.contigs_index  ;

#Map reads
basen=coli_muted_n_30_genome_
suffixn=_reads.fasta
for i in $(seq 0 30); do
 bowtie2 -f --non-deterministic --threads 8  --rg-id "read$i" --rg "SM:read$i" \
  --rg "PL:Illumina" --rg "LB:simumima" -x NAR30A_ref.contigs_index  -U ${basen}${i}${suffixn} \
  |  samtools view -bS - >  NAR30A_${i}_bw2.bam;
 samtools sort  -m 25000000000 NAR30A_${i}_bw2.bam NAR30A_${i}_bw2.sorted.bam
 mv NAR30A_${i}_bw2.sorted.bam.bam NAR30A_${i}_bw2.bam
 samtools index NAR30A_${i}_bw2.bam
 samtools flagstat NAR30A_${i}_bw2.bam > NAR30A_${i}_bw2.flagstat
done

#Variant calling
samtools faidx coli_muted_ref.contigs.fa
java -Xmx4g -jar ./CreateSequenceDictionary.jar  R=coli_muted_ref.contigs.fa  O=coli_muted_ref.contigs.dict
gpar="-I coli_muted_0_bw2.bam"
for i in $(seq 1 30); do
 gpar="$gpar -I coli_muted_${i}_bw2.bam"
 java -Xmx8g -jar /softs/local/GATK/2.8.1/GenomeAnalysisTK.jar -R coli_muted_ref.contigs.fa -T UnifiedGenotyper \
 -glm BOTH ${gpar} -o coli_muted_both_${i}.vcf
done
```

Table 1 Commands used for calling SNPs and indels from 2 to 30 E. Coli read sets (coli_muted_n_30_genome_i_reads.fasta for i in $[1, 30]$) with *DiscoSnp++*, *cortex* or the hybrid approach.

DiscoSnp++

```
run_discoSnp++.sh -r "humch1_00096_reads.fasta humch1_00100_reads.fasta" -P 4 -D 10
```

cortex

```
#Prepare data
for i in 096 100; do
 cortex_var_31_c2 --se_list data$i.txt --kmer_size 31 --mem_height 25 --mem_width 100 \
  --dump_binary genome_$i\_reads.ctx  --sample_id sample$i --remove_low_coverage_supernodes 3
done

#Prepare parameters
for i in 096 100; do
 echo  "genome_$i\_reads.ctx" > ind$i\_ctx
 echo ind$i\_ctx >> my_cortex_list_ctx
done

#Detect variants
cortex_var_31_c2 --colour_list my_cortex_list_ctx --kmer_size 31 --mem_height 25 --mem_width 100 \
 --detect_bubbles1 0,1/0,1 --output_bubbles1 my_res_cortex --remove_low_coverage_supernodes 3
```

Hybrid approach

```
#Assembly
SOAPdenovo-63mer pregraph -s soap.config  -o soaphum -K $k -d 5
SOAPdenovo-63mer contig -g soaphum

#Remove contigs whose size is < 100:
./filter_fasta_by_length.py soaphum.contig hum_ref.contigs.fa 100
mv  soaphum.contig  hum_ref.contigs.fa

#Align reads
${path_bw2}bowtie2-build -f hum_ref.contigs.fa hum_ref.contigs_index
for i in 096 100; do
 bowtie2 -f --non-deterministic --threads 8  --rg-id "read$i" --rg "SM:read$i" --rg "PL:Illumina" \
 --rg "LB:simumima" -x hum_ref.contigs_index  -U /WORKS/tests_human/data/humch1_00$i\_reads.fasta \
 |  ${path_sam}samtools view -bS - >  hum_${i}_bw2.bam;
 samtools sort  -m 25000000000 hum_${i}_bw2.bam hum_${i}_bw2.sorted.bam
 mv hum_${i}_bw2.sorted.bam.bam hum_${i}_bw2.bam
 samtools index hum_${i}_bw2.bam
 samtools flagstat hum_${i}_bw2.bam > hum_${i}_bw2.flagstat
done

#Variant calling
samtools faidx hum_ref.contigs.fa
java -Xmx4g -jar ./CreateSequenceDictionary.jar  R=${ref}  O=hum_ref.contigs.dict
time ~grizk/bin/memusage java -Xmx62g -jar /softs/local/GATK/2.8.1/GenomeAnalysisTK.jar \
-R ${ref} \
-T HaplotypeCaller \
-I ${prefix}_096_bw2.bam -I ${prefix}_100_bw2.bam \
-o ${prefix}_both_HaplotypeCaller.vcf
```

Table 2 Commands used for calling SNPs and indels from two human chromosome 1 read sets (humch1_00096_reads.fasta humch1_00100_reads.fasta) with *DiscoSnp++*, *cortex* or the hybrid approach.