Index

Variation detection based on Illumina sequencing

In this part, drill on variation is included. Variation detection based on Illumina sequencing result basically contains several steps: 1) filter the raw data; 2) map the filtered data to the reference genome; 3) summarize the mapping result; 4) variation detection. These processes will be introduced separately in the following sections. All the data used here were simulated Illumina sequencing result from a designed reference genome with many know single nucleotide polymorphisms. Raw data are in the directory of "/home/variation_detection/00.RawData". And for convenience, all the command lines were included in shell scripts in respective directories.

> For all the software/tools we presented, we did not introduce all the parameters. And we can find the description for each parameter by two approaches:
>
> 1) For most of the software, you can just type in the help command in order to get all the details for all the parameters. For example, "cut --help", "awk -h" or without any input "soap" or "bwa";
> 2) You can find the manual distributed with the software. This is also the case for data format. You can find the data format description in the manual.

Raw data filtering

The raw data from the sequencer should first be filtered in order to get clean data. For Illumina sequencing platforms (or other platforms), there are several possible kinds of reads which should be filtered, including a) reads which were duplicated in the PCR process, b) reads which were with low quality. As those reads would affect the variation detection, we need to filter those reads.

First go to the directory by the following command:

cd /home/variation_calling/ 01.DataFiltering

Then in the directory of "01.bin", there are software which can be used to filter the data. We can use the software to filter the low quality data and the duplicated reads by the following commands:

01.bin/Filter_quality -y -z -w 10 -B 40 -l 140 -a 0 -b 0 -c 0 -d 0 ../00.RawData/SRR191857_1.fastq.gz ../00.RawData/SRR191857_2.fastq.gz Step1.filtering.stat SRR191857_1.clean.fq.gz SRR191857_2.clean.fq.gz

01.bin/Filter_duplication SRR191857_1.clean.fq.gz SRR191857_2.clean.fq.gz SRR191857_1.dup.clean.fq.gz SRR191857_2.dup.clean.fq.gz Step2.dup.filtering.stat

After executing these commands, clean data (SRR191857_1.dup.clean.fq.gz and SRR191857_2.dup.clean.fq.gz) are obtained and will be further used. And all these commands are included in the shell script "filter_data.sh", so just type "sh filter_data.sh" will also generate the same results.

```
[liuxin@login-0-14 01.DataFiltering]$ cd /home/variation_calling/01.DataFiltering
[liuxin@login-0-14 01.DataFiltering]$ ls
01.bin
[liuxin@login-0-14 01.DataFiltering]$ 01.bin/Filter_quality -y -z -w 10 -B 40 -l 140 -a 0 -b 0 -c 0 -d 0 ../00.RawData/SRR191857_1.fas
tq.gz ../00.RawData/SRR191857_2.fastq.gz Step1.filtering.stat SRR191857_1.clean.fq.gz SRR191857_2.clean.fq.gz
76      76      76      76
start to read fq.gz file
[liuxin@login-0-14 01.DataFiltering]$ 01.bin/Filter_duplication SRR191857_1.clean.fq.gz SRR191857_2.clean.fq.gz SRR191857_1.dup.clean.
fq.gz SRR191857_2.dup.clean.fq.gz Step2.dup.filtering.stat
[liuxin@login-0-14 01.DataFiltering]$ ls
01.bin                  SRR191857_1.dup.clean.fq.gz  SRR191857_2.dup.clean.fq.gz  Step2.dup.filtering.stat
SRR191857_1.clean.fq.gz  SRR191857_2.clean.fq.gz      Step1.filtering.stat
[liuxin@login-0-14 01.DataFiltering]$
```

Mapping

After getting the filtered data, we need to map the reads to the reference genome. There are different software to map short reads to the reference genome. Here we will use two of them, soap and bwa.

Mapping by soap

First to use soap to do the mapping, we need build index for the reference in advance. Use the following command we can build index for the reference genome, after we go to the directory "/home/variation_calling/02.Mapping/02.SOAP":

cd /home/variation_calling/02.Mapping/02.SOAP

./2bwt-builder ../../00.RawData/Ostreococcus_tauri.fa

```
[liuxin@login-0-14 02.SOAP]$ cd /home/variation_calling/02.Mapping/02.SOAP
[liuxin@login-0-14 02.SOAP]$ ls
2bwt-builder  soap2.20
[liuxin@login-0-14 02.SOAP]$ ./2bwt-builder ../../00.RawData/Ostreococcus_tauri.fa
Parsing FASTA file..
Finished. Parsed 20 sequences.
Elapsed time = 0.63 s

Building Look-Up..
Finished.
Elapsed time = 7.89 s

Building BWT..
Finished constructing BWT in 77 iterations.  Elapsed time = 3.13 s

Saving BWT..
Finished saving BWT.  Elapsed time = 0.07 s

Building Reversed BWT..
Finished constructing Reversed BWT in 77 iterations.  Elapsed time = 3.10 s

Saving BWT..
Finished saving BWT.  Elapsed time = 0.06 s

Loading BWT...
Finished loading BWT.  Elapsed time = 0.04 s

Building SA value...
Finished building SA value.  Elapsed time = 1.07 s

Building High-Occ Hash Table...
Finished.
Elapsed time = 0.69 s

Building SA index...
Finished building SA index.  Elapsed time = 3.42 s

Index building is completed.
Total elapsed time = 20.10 s
```

Then we can do the mapping by soap using the following command:

./soap2.20 -a ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz -b ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz -D ../../00.RawData/Ostreococcus_tauri.fa.index -m 200 -x 1500 -o paired_mapped.soap -2 single_mapped.soap -u unmapped_reads.fa

```
[liuxin@login-0-14 02.SOAP]$ ./soap2.20 -a ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz -b ../../01.DataFiltering/SRR191857_2.du
p.clean.fq.gz -D ../../00.RawData/Ostreococcus_tauri.fa.index -m 200 -x 1500 -o paired_mapped.soap -2 single_mapped.soap -u unmapped_r
eads.fa

Begin Program SOAPaligner/soap2
Wed Nov 20 22:53:16 2013
Reference: ../../00.RawData/Ostreococcus_tauri.fa.index
Query File a: ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz
Query File b: ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz
Output File: paired_mapped.soap
             single_mapped.soap
             unmapped_reads.fa
Load Index Table ...
Load Index Table OK
Begin Alignment ...
131072 ok    5.58 sec
262144 ok    5.88 sec
393216 ok    5.90 sec
524288 ok    5.99 sec
655360 ok    5.76 sec
786432 ok    5.41 sec
917504 ok    5.45 sec
1048576 ok    5.59 sec
1179648 ok    5.60 sec
1310720 ok    5.25 sec
1441792 ok    4.77 sec
1572864 ok    4.83 sec
1703936 ok    5.01 sec
1835008 ok    5.46 sec
1966080 ok    5.07 sec
2097152 ok    4.78 sec
2228224 ok    4.80 sec
2357384 ok    4.74 sec
Total Pairs: 1178692 PE
Paired:      298647 (25.34%) PE
Singled:     551793 (23.41%) SE
Total Elapsed Time:          133.64
      - Load Index Table:      8.36
      - Alignment:           125.28
```

After executing these commands (also included in Step1.Mapping.sh in this directory), the mapping results were included in “paired_mapped.soap” and “single_mapped.soap” in the format of soap. Then we can split the mapping result according to chromosomes and sort the mapping result according to the mapping positions. And to make those mapping results ready for variation

detection, further processing including split according to chromosomes and sort according to mapping positions should be done. The process can be finished using the script "Step2.Processing.sh", which will generate the file "Chr1.sort" with the chromosomes split and sorted.

```
[liuxin@login-0-14 02.SOAP]$ cat paired_mapped.soap single_mapped.soap |grep -w Chr01|sort -k 9 -n >Chr01.sort
[liuxin@login-0-14 02.SOAP]$ les Chr01.sort
SRR191857.823469        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAGACCCTAACCCCTAAACCCTAAACCCTAA    \[XUVVVTUTTTWVZ]VZ[\XTT[WVZ][]
SRR191857.1170210       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    V[UX[\W]YXWXTZY[2XZ\[U\YVU[WY]
SRR191857.1276749       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    [W\Y[Z\[YXV\Y\V]\W\]UZTXYVYUVY
SRR191857.847483        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAGACCCTAAACCCTAAACCCTAAACCCTAA    ZV]V[\[XXW\V\YTT[[WU\]VUVTXU]V
SRR191857.825465        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAGACCCTAAACCCTAAACCCTAAACCCTAA    VY]]VYWYZ[ZTYZXX]U]\WT]\Y]VZV\
SRR191857.1393577       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    UWZYV\]ZVZX]XYU[TUZ[U]U\UXW[YZ
SRR191857.514311        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    UWV[VVXUTW[X[]Y]XVYWWYWU[TU]ZX
SRR191857.945614        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACTCTAA    ZUTT]X]Y[XWUY[VUUZWU[]VYUZVVUV
SRR191857.997197        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    VWTZTU[VXW[VXUWWXTZY]WVVUWVYYT
SRR191857.512472        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTCAACCCTAAACCCTAA    YV\]YW\\\[[]UWTWYTY]UTZZX[TVUU
SRR191857.1419855       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    [\YZWYTV\]\U]U][[\][TVTYZWW]V]
SRR191857.1423439       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    [XXWTTXV\VZUZWU][\Z[YYXWWY]VZV
SRR191857.357671        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    UX]X[VTVVXTTTU]YWUZYTV\ZWZXY[
SRR191857.935075        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    \W[V\Z[WZYZZY\ZUUW[TYUUV[TTV[U
SRR191857.948538        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    TUT]YW]ZZWZ]WVX[[[YZT]]VVZ[]UU
SRR191857.1042632       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCATAA    XZWZVWTYXYU]YYZX[ZWW]YU]T[WZUW
SRR191857.1214103       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    UX\\Y]Z]WXY]UUVTZVYYZXWW\[YY\[
SRR191857.708258        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAA    Y[V]XTUVX[[]Y[]YZ[YWZ\TYUWXZUV
SRR191857.437162        ACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAATCTAAATGACG    ADEIGCGFAA@@HCDGEIFADBDGHIUVXX
SRR191857.1354342       CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACGAACC    XW]WUX\YY[ZZY\XUZT[[[\]U]Z\]\W
SRR191857.533056        CCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACCCTAAACGAACC    [YX]WZTZ]\VZYYU]TT]ZY\[UXZTXYT
SRR191857.1390749       TTACACGATTCGAGTGTCATCGTCTTTTCAACGACGCTTGCGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCC    ]TY[]V[Z\YXT]]UZVT\ZVTYWT[U]VX
SRR191857.1161553       TCATCGTCTTTTCAACGACGCTTGCGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACA    EAGGIBIIDEICIDDHCHBHGFCGFFV]]]
SRR191857.796409        CTTTTCAACGACGCTTGCGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCAT    @ECAHIFADIBBB@G@AFHAFDDHEH[UWY
SRR191857.840590        TTTCAACGACGCTTGCGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCATCC    [V[V\]]TXZYZW[ZVT]UZZY]T[]]Z]U
SRR191857.662612        TTGCGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCATCCGTTTTTATCGGC    CIGGH@F@EACICBBF@CGCHFBHIB\Y]W
SRR191857.808351        CGCTCCGAAAATGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCATCCGTTTTTATCGGCGCC    C@FIIICFHGEEAFHE@GIIGAIACET\Z\
SRR191857.1390749       TGCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCATCCGTTTTTATCGGCGCCGAGCCCTCAAA    AFEADAEIEBFAFDFAIDFBCHF@GDXYXT
SRR191857.549559        GCTCGCGCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCATCATCCGTTTTTATCGGCGCCGAGCCCTCAAAA    CBEIAAHFFIC@ACFDBEH@F@BFAHW\YY
SRR191857.1228652       GCGCTGAGCTACGGACCCTGTTTTGTATTCGACATCAGAATCCGTTTTTATCGGCGCCGAGCCCTCAAAAACAGCC    EA@DIGBFHHFFHGBGEFHDECAFFG]YV[
SRR191857.626051        TGTTTTGTATTCGACATCATCATCCGTTTTTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCGCCGTGATGCTC    @DDIDHDFE@BIFIBCIEHBDDEB@IT[TT
SRR191857.959206        TGTTTTGTATTCGACATCATCATCCGTTCTTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCGCCGTGATGCTC    BADBDDC@ACAGBAFAFBABIIDIIIZV]Y
SRR191857.1415958       ACATCATCATCCGTTTTTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCGCCGTGATGCTCGGAGAGATATTTC    YVXXYT[V\[\U\T[TU]]XYWU\ZYZUTU
SRR191857.1336138       CATCATCATCCGTTTTTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCGCCGTGATGCTCGGAGAGATATTTCC    XZYT\]TYU]TUUZ[ZXV[\ZXU[Z\]ZX[
SRR191857.1253307       CATCCGTTTTTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCGCCGTGATGCTCGGAGAGATATTTCCACCCTT    EGFBEFCCCFI@BGE@EEH@BIBGDEY]TW
SRR191857.1005863       TTATCGGCGCCGAGCCCTCAAAAACAGCCGCGCGCCCCTGATGCTCGGAGAGATATTTCCACCCTTCGGCAAAGA    FIICFIGI@EADBGBA@DIDCEIBAEY[VY
SRR191857.498897        CAAAGAAACTGGGCTCTCGCGTTTGACGAGTTCGGGCTCTCGCACTGCCGCTGTCGCTCCATCTATTTTCACCCTG    \ZTXYUVWTXV[]\]\XXV]][\[WV[V[Z
```

Mapping by bwa

In order to use bwa for mapping, we should also build index for both the reads and reference using specific bwa functions.

First go the directory,

cd /home/variation_calling/02.Mapping/01.bwa

Then to build index for the reference genome:

./bwa index ../../00.RawData/Ostreococcus_tauri.fa

```
[liuxin@login-0-14 01.bwa]$ cd /home/variation_calling/02.Mapping/01.bwa
[liuxin@login-0-14 01.bwa]$ ls
bwa
[liuxin@login-0-14 01.bwa]$ ./bwa index ../../00.RawData/Ostreococcus_tauri.fa
[bwa_index] Pack FASTA... 0.22 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 11.43 seconds elapse.
[bwa_index] Update BWT... 0.12 sec
[bwa_index] Pack forward-only FASTA... 0.18 sec
[bwa_index] Construct SA from BWT and Occ... 3.26 sec
[main] Version: 0.6.2-r126
[main] CMD: ./bwa index ../../00.RawData/Ostreococcus_tauri.fa
[main] Real time: 15.765 sec; CPU: 15.213 sec
```

Then to build the index for the raw reads fastq data:

./bwa aln -n 3 ../../00.RawData/Ostreococcus_tauri.fa ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz >1.dup.clean.fqi

./bwa aln -n 3 ../../00.RawData/Ostreococcus_tauri.fa ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz >2.dup.clean.fqi

```
[liuxin@login-0-14 01.bwa]$ ./bwa aln -n 3 ../../00.RawData/Ostreococcus_tauri.fa ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz >
1.dup.clean.fqi
[bwa_aln_core] calculate SA coordinate... 18.42 sec
[bwa_aln_core] write to the disk... 0.03 sec
[bwa_aln_core] 262144 sequences have been processed.
[bwa_aln_core] calculate SA coordinate... 18.48 sec
[bwa_aln_core] write to the disk... 0.04 sec
[bwa_aln_core] 524288 sequences have been processed.
[bwa_aln_core] calculate SA coordinate... 18.54 sec
[bwa_aln_core] write to the disk... 0.04 sec
[bwa_aln_core] 786432 sequences have been processed.
[bwa_aln_core] calculate SA coordinate... 18.66 sec
[bwa_aln_core] write to the disk... 0.04 sec
[bwa_aln_core] 1048576 sequences have been processed.
[bwa_aln_core] calculate SA coordinate... 9.41 sec
[bwa_aln_core] write to the disk... 0.02 sec
[bwa_aln_core] 1178692 sequences have been processed.
[main] Version: 0.6.2-r126
[main] CMD: ./bwa aln -n 3 ../../00.RawData/Ostreococcus_tauri.fa ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz
[main] Real time: 86.596 sec; CPU: 86.536 sec
[liuxin@login-0-14 01.bwa]$ ./bwa aln -n 3 ../../00.RawData/Ostreococcus_tauri.fa ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz >
2.dup.clean.fqi
[bwa_aln_core] calculate SA coordinate... 3.71 sec
[bwa_aln_core] write to the disk... 0.02 sec
[bwa_aln_core] 262144 sequences have been processed.
```

After getting the index files, we can do the mapping:

./bwa sampe -a 1500 ../../00.RawData/Ostreococcus_tauri.fa 1.dup.clean.fqi 2.dup.clean.fqi ../../01.DataFiltering/SRR191857_1.dup.clean.fq.gz ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz >mapping.sam

```
[liuxin@login-0-14 01.bwa]$ ./bwa sampe -a 1500 ../../00.RawData/Ostreococcus_tauri.fa 1.dup.clean.fqi 2.dup.clean.fqi ../../01.DataFi
ltering/SRR191857_1.dup.clean.fq.gz ../../01.DataFiltering/SRR191857_2.dup.clean.fq.gz >mapping.sam
[bwa_sai2sam_pe_core] convert to sequence coordinate...
[infer_isize] fail to infer insert size: too few good pairs
[bwa_sai2sam_pe_core] time elapses: 1.84 sec
[bwa_sai2sam_pe_core] changing coordinates of 0 alignments.
[bwa_sai2sam_pe_core] align unmapped mate...
[bwa_sai2sam_pe_core] time elapses: 0.02 sec
[bwa_sai2sam_pe_core] refine gapped alignments... 0.42 sec
[bwa_sai2sam_pe_core] print alignments... 1.47 sec
[bwa_sai2sam_pe_core] 262144 sequences have been processed.
[bwa_sai2sam_pe_core] convert to sequence coordinate...
[infer_isize] (25, 50, 75) percentile: (207, 218, 229)
[infer_isize] low and high boundaries: 163 and 273 for estimating avg and std
[infer_isize] inferred external isize from 86080 pairs: 218.582 +/- 16.671
[infer_isize] skewness: -0.399; kurtosis: 0.563; ap_prior: 3.02e-05
[infer_isize] inferred maximum insert size: 322 (6.22 sigma)
[bwa_sai2sam_pe_core] time elapses: 3.82 sec
[bwa_sai2sam_pe_core] changing coordinates of 1162 alignments.
[bwa_sai2sam_pe_core] align unmapped mate...
[bwa_paired_sw] 83673 out of 87297 Q17 singletons are mated.
[bwa_paired_sw] 266 out of 789 Q17 discordant pairs are fixed.
[bwa_sai2sam_pe_core] time elapses: 20.65 sec
[bwa_sai2sam_pe_core] refine gapped alignments... 0.70 sec
[bwa_sai2sam_pe_core] print alignments... 1.87 sec
[bwa_sai2sam_pe_core] 524288 sequences have been processed.
[bwa_sai2sam_pe_core] convert to sequence coordinate...
[infer_isize] (25, 50, 75) percentile: (207, 218, 230)
[infer_isize] low and high boundaries: 161 and 276 for estimating avg and std
[infer_isize] inferred external isize from 114132 pairs: 218.670 +/- 16.878
[infer_isize] skewness: -0.438; kurtosis: 0.666; ap_prior: 2.85e-05
[infer_isize] inferred maximum insert size: 324 (6.22 sigma)
[bwa_sai2sam_pe_core] time elapses: 4.35 sec
[bwa_sai2sam_pe_core] changing coordinates of 1661 alignments.
[bwa_sai2sam_pe_core] align unmapped mate...
[bwa_paired_sw] 57993 out of 61681 Q17 singletons are mated.
[bwa_paired_sw] 369 out of 1079 Q17 discordant pairs are fixed.
[bwa_sai2sam_pe_core] time elapses: 15.03 sec
[bwa_sai2sam_pe_core] refine gapped alignments... 0.75 sec
```

Thus we get the final mapping result “sample1.sam” in the sam format (these commands are in “Step1.Mapping.sh”). As what we did for the soap mapping result, we also need to process this result in order for further use. “samtools” is always used to treat sam files. We first change the sam format file into bam format file which is much smaller in file size and sort it. Then, we can build index for the sorted bam file (these commands are also included in “Step02.treat_sam.sh”).

./samtools view -S mapping.sam -b|./samtools sort - mapping.sort

./samtools index mapping.sort.bam

```
[liuxin@login-0-14 01.bwa]$ ./samtools view -S mapping.sam -b|./samtools sort - mapping.sort
[samopen] SAM header is present: 20 sequences.
[liuxin@login-0-14 01.bwa]$ ./samtools index mapping.sort.bam
[liuxin@login-0-14 01.bwa]$ ls
1.dup.clean.fqi  Step1.Mapping.sh     bwa          mapping.sort.bam      samtools
2.dup.clean.fqi  Step2.Processing.sh  mapping.sam  mapping.sort.bam.bai
```

Summarize the mapping result

After getting the mapping result, it would be important to summarize the mapping results (especially about sequencing depth, sequencing coverage and mapping rate), in order to make sure the sequencing as well as mapping analysis is good. For the soap format mapping result, we can use the "soapcoverage" function to summarize the mapping result (we need first to go to the directory of "/home/variation_calling/03.SummarizeMapping"; and this command is also in the script of "SummarizingSOAP.sh"):

```
./soap.coverage -cvg    -refsingle      ../00.RawData/Ostreococcus_tauri.fa     -
i     ../02.Mapping/02.SOAP/paired_mapped.soap     ../02.Mapping/02.SOAP/single_mapped.soa
p -depthsingle    1.depth
```

```
[liuxin@login-0-14 03.SummarizeMapping]$ cd /home/variation_calling/03.SummarizeMapping
[liuxin@login-0-14 03.SummarizeMapping]$ ./soap.coverage -cvg    -refsingle      ../00.RawData/Ostreococcus_tauri.fa     -i      ../02
.Mapping/02.SOAP/paired_mapped.soap     ../02.Mapping/02.SOAP/single_mapped.soap -depthsingle    1.depth

SOAP.coverage
    Version: 2.7.7
Complied at: Dec 31 2009 14:58:44
     Author: RuiBang Luo
     E-mail: luoruibang@genomics.org.cn

This utility can calculate sequencing coverage or physical coverage as well as duplication rate
and details of specific block for each segments and whole genome by using SOAP, Blat, Blast, BlastZ,
mummer and MAQ aligement results with multi-thread. Gzip file supported.


Parameters List:
-cvg
-refsingle ../00.RawData/Ostreococcus_tauri.fa
-i ../02.Mapping/02.SOAP/paired_mapped.soap ../02.Mapping/02.SOAP/single_mapped.soap
-depthsingle 1.depth
# End of parameters list

Picking out segments from reference file...
Number of segments:        20
General output (-o) undefined, disabled!
Shadowing Map...
Mutex Lock created: 20
Building Memory Blocks...
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
Summarizing Coverage...
1/2: ../02.Mapping/02.SOAP/paired_mapped.soap
2/2: ../02.Mapping/02.SOAP/single_mapped.soap
Calculating Coverage...
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
Output Coverage to files (Text)...
```

Then we can know that the coverage is about 88% by soap and the sequencing depth for each loci is in the "1.depth" fasta like file.

For the bwa mapping result in the sam or bam format, we have many tools including "samtools" to summarize the mapping result. Use "samtools", we can easily summarize the mapping result by (also in "SummarizingBAM.sh"):

```
./samtools flagstat ../02.Mapping/01.bwa/mapping.sort.bam
```

```
[liuxin@login-0-14 03.SummarizeMapping]$ ./samtools flagstat ../02.Mapping/01.bwa/mapping.sort.bam
2357384 in total
0 QC failure
0 duplicates
1478268 mapped (62.71%)
2357384 paired in sequencing
1178692 read1
1178692 read2
1274648 properly paired (54.07%)
1279805 with itself and mate mapped
198463 singletons (8.42%)
3312 with mate mapped to a different chr
2679 with mate mapped to a different chr (mapQ>=5)
```

Variation detection

Now we can use different software to detect different variations using those mapping results. Here we just show examples of how to use soapsnp and samtools to detect SNPs.

To use soapsnp detect SNPs, we can use the following command (in "SoapSnp.sh"):

```
./soapsnp -i ../02.Mapping/02.SOAP/Chr01.sort -d ../00.RawData/Chr01.fa -o Chr01.cns -L 100

./FilterSNP.pl -CnsFile Chr01.cns -MinQual 25 -MaxRept 1.8 -MinDist 5 -MinDept 10 -MaxDept 30 -MinPval 0.05 >Chr1.snp
```

```
[liuxin@login-0-14 04.VariationCalling]$ ./soapsnp -i ../02.Mapping/02.SOAP/Chr01.sort -d ../00.RawData/Chr01.fa -o Chr01.cns -L 100
Reading Chromosome and dbSNP information Done.
Correction Matrix Done!
Processing Chr01
Consensus Done!
[liuxin@login-0-14 04.VariationCalling]$ ./FilterSNP.pl -CnsFile Chr01.cns -MinQual 25 -MaxRept 1.8 -MinDist 5 -MinDept 10 -MaxDept 30
 -MinPval 0.05 >Chr1.snp
[liuxin@login-0-14 04.VariationCalling]$ ls
Chr01.cns  Chr1.snp  FilterSNP.pl  SoapSnp.sh  samtools  soapsnp
```

We can also detect variations using samtools by the following commands ("samtools_pileup.sh"):

```
./samtools pileup -c -2 -v -f ../00.RawData/Ostreococcus_tauri.fa ../02.Mapping/01.bwa/mapping.sort.bam >bwa.variations
```

```
[liuxin@login-0-14 04.VariationCalling]$ ./samtools pileup -c -2 -v -f ../00.RawData/Ostreococcus_tauri.fa ../02.Mapping/01.bwa/mappin
g.sort.bam >bwa.variations
```