SUPPLEMENTAL TEXT:

1. Data types used in COSINE

Fastq files:

Raw sequencing data to be processed of tumor-normal paired sample, usually, a minimum sequencing depth is 60x for whole exome sequencing (WES) and 30x for whole genome sequencing (WGS) for clonal evolution analysis.

Variant call format (VCF) files:

The VCF file generally records the detailed information of the SNP in the genome. This file is usually generated by varscan2/mutect2 or strelka, and generally used as the input data of the tumor evolution analysis software in the computing platform.

Other files:

Other types of files can also be processed with COSINE, mainly including input files of 12 kinds of analysis software, or the intermediate generated files of each software. The formats of these files include not only txt, csv, yaml and so on.

2. Variation calling process

Step-1: Alignment and sorting of sequencing data to reference sequences

In this step, raw clean sequencing data will alignment to the reference genome by BWA software package. The alignment results are saved in sam format, but the sam file is too large, and samtools was used to convert the sam file to bam file.

```
bwa mem -t 24 -M -Y -R \
"@RG\tID:${sampleID}\tPL:illumina\tLB:WGS\tSM:${sampleID}" \
$reference $fasgq1 $fastq2 > ${sampleID}.WGS.sam
```

The comparison results obtained from BWA files need to be further processed by the samtools. It can sort, merge and index the bam files, and convert sam files into bam files. *view* function of samtools for converting the sam file into bam files with .more than 6 times less in size, and *sort* function for sorting the bam files, -@ parameter was set the number of number of CPU used for parallel computing, -o parameter was set the output file name:

```
samtools view -Sb ${sampleID}.WGS.sam > ${sampleID}.WGS.bam
samtools sort -@ 24 -o ${sampleID}.WGS.sorted.bam ${sampleID}.WGS.bam
```

Step-2: Base calibration via GTAK toolkits

The GATK can mark duplication reads in the alignment bam file, which caused by PCR amplification and other reasons. In the specific command, *-I* set the input file name, *-O* controls the output file name, and -REMOVE_DUPLICATES represents whether to delete the repeated sequence.

```
gatk MarkDuplicates -I ${sampleID}.WGS.sorted.bam \
-O ${sampleID}.WGS.sorted.markdup.bam \
-M ${sampleID}.WGS.sorted.markdup.txt \
```

```
-REMOVE_DUPLICATES true
```

The BQSR process in the GATK is contain two steps. First is using the known mutation information to filter the sequencing result data, the software can generate an intermediate file in the table format. Second is using the table file, the sequencing data and the reference, the software can make the final removal of untrusted mutation sites in sequencing data.

```
gatk BaseRecalibrator -R $reference \
-I ${sampleID}.WGS.sorted.markdup.bam \
--known-sites $known_indel1 --known-sites $known_snp \
--known-sites $known_indel2 -O ${sampleID}.table

gatk ApplyBQSR --bqsr-recal-file ${sampleID}.table \
-R $reference -I ${sampleID}.WGS.sorted.markdup.bam \
-O ${sampleID}.WGS.sorted.markdup.bqsr.bam
```

Step-3: Single nucleotide variations (SNV) calling

In this step, we use varscan2 or mutect2 in the GATK to detect the SNV in the adjused BAM file.

```
#mutect2:
gatk Mutect2 -R $reference -I ${sampleID}.WGS.sorted.markdup.bqsr.bam\
-L $interval_list -O ${sampleID}.WGS.mutect2.vcf

gatk FilterMutectCalls -V ${sampleID}.WGS.mutect2.vcf \
-O ${sampleID}.WGS.somatic.vcf -R $reference

#Varscan2:
varscan somatic < (samtools mpileup \
--no-BAQ -f $reference \
$normal_contral.sorted.markdup.bam \
${sampleID}.WGS.sorted.markdup.bam) \
$output_dir --mpileup 1 --output-vcf --min-coverage-tumor 15 \
--min-coverage-norml 12 --somatic-p-value 0.01

varscan processSomatic ${sampleID}.WGS.snp.vcf
varscan processSomatic ${sampleID}.WGS.indel.vcf
```

Figure 1 is a result file, which is include of SNV and indel mutation detected by varscan2. In the following chapters, we will filter out the somatic variation from the output files as the analysis sample.

```
606384   chr10   132622895   .   C   T   .   PASS    DP=83;SS=1;SSC=2;GPV=2.4559e-07;SPV=0.60039 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:5
606385   chr10   132622985   .   G   C   .   PASS    DP=52;SS=1;SSC=2;GPV=9.8643e-08;SPV=0.61197 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:2
606386   chr10   132625447   .   G   A   .   PASS    DP=42;SS=1;SSC=1;GPV=2.607e-06;SPV=0.7348   GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:2
606387   chr10   132625840   .   CTGTGTG C   .   PASS    DP=53;SS=1;SSC=6;GPV=5.7598e-07;SPV=0.20045 GT:GQ:DP:RD:AD:FREQ:DP4 0/1
606388   chr10   132625974   .   T   C   .   PASS    DP=32;SS=1;SSC=2;GPV=1.3089e-07;SPV=0.57423 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:2
606389   chr10   132626319   .   C   T   .   PASS    DP=43;SS=1;SSC=0;GPV=5.2508e-08;SPV=0.94505 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:2
606390   chr10   132626378   .   G   A   .   PASS    DP=47;SS=1;SSC=0;GPV=6.1512e-28;SPV=1   GT:GQ:DP:RD:AD:FREQ:DP4 1/1:.:30:0:
606391   chr10   132626561   .   C   T   .   PASS    DP=53;SS=1;SSC=5;GPV=8.7266e-12;SPV=0.28525 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:3
606392   chr10   132628463   .   CGG C   .   PASS    DP=33;SS=1;SSC=6;GPV=0.0038123;SPV=0.22021  GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:1
606393   chr10   132629208   .   T   G   .   PASS    DP=52;SS=1;SSC=0;GPV=6.3169e-31;SPV=1   GT:GQ:DP:RD:AD:FREQ:DP4 1/1:.:28:0:
606394   chr10   132629528   .   A   G   .   PASS    DP=61;SS=1;SSC=1;GPV=8.637e-13;SPV=0.79121  GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:4
606395   chr10   132630615   .   A   G   .   PASS    DP=41;SS=1;SSC=1;GPV=1.6028e-05;SPV=0.66419 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:2
606396   chr10   132636163   .   A   ATGTGTG .   PASS    DP=57;SS=1;SSC=2;GPV=1.364e-06;SPV=0.57992  GT:GQ:DP:RD:AD:FREQ:DP4 0/1
606397   chr10   132636996   .   A   T   .   PASS    DP=71;SS=1;SSC=0;GPV=2.68349e-42;SPV=1  GT:GQ:DP:RD:AD:FREQ:DP4 1/1:.:39:0:
606398   chr10   132637063   .   G   A   .   PASS    DP=65;SS=1;SSC=4;GPV=8.9089e-11;SPV=0.3858  GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:3
606399   chr10   132640399   .   C   T   .   PASS    DP=76;SS=1;SSC=2;GPV=7.6319e-16;SPV=0.59075 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:3
```

Figure 1 The results of varscan2 are saved in VCF format

Step-4: Copy number variation calling

In addition to SNP variation, genome variation includes copy number variation and structure variation. In COSINE, sclust is used to detect the variation of copy number.

```
Sclust bamprocess -t ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -n ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -o ${sampleID} -part 2 -build hg38 -r chr1

Sclust bamprocess -t ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -n ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -o ${sampleID} -part 2 -build hg38 -r chr2
...##running for all 24 chorosomes
Sclust bamprocess -i ${sampleID}. -o ${sampleID}.
Sclust cn -rc ${sampleID}_rcount.txt -snp ${sampleID}_snps.txt \
-vcf ${sclust}.input.vcf -o ${sampleID} --ns 1000
```

Figure 2 shows the copy number variation detected by the Sclust, and the purity and ploidy of the tumor-normal paired genomic data. In addition, copy number (brown) and minor allele copy number (green) of each chromosome were shown.

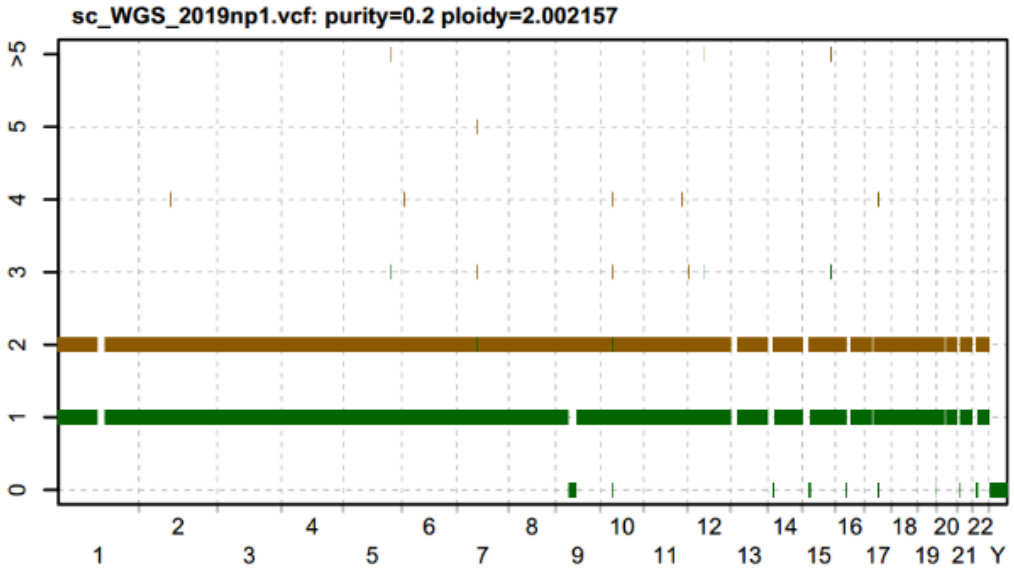

Figure 2 The result file of the Sclust running *cn* function

3. Preprocess of subclonal inference

here is a workflow for convert VCF file to different subclonal inference method.

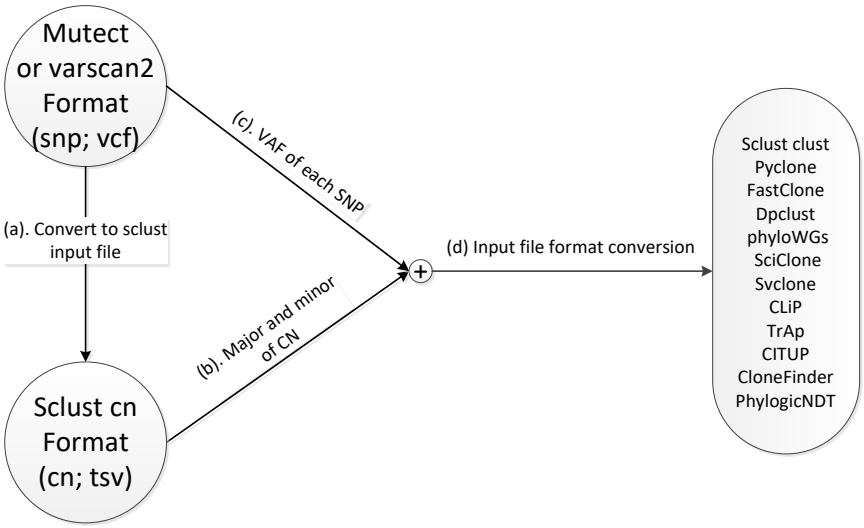

Figure 3 Method of format conversion

Step-1: Convert the SNP information into the input format of Sclust

On the basis of input file conversion, the software of mutation detection and evolutionary analysis can dock without data loss, and the analysis process of tumor cell information is improved. The platform can use the same data to evaluate the performance of different software, and provides reliable and stable running results.

The function of the script named *varscanTosclust.py* is to convert varscan2 output file to sclust input file. *-i* represents varscan software indel detection result; *-s* represents varscan software snp detection result; *-o* means the file name of conversion result.

```
python $map_dir/bash/varscanTosclust.py -i $varscan2.out.indel.vcf \
                                        -s $varscan2.out.snp.vcf \
                                        -o ${sclust}.input.vcf
```

The function of the script named *mutectTosclust.py* is to convert mutect2 output file to sclust input file. *-i* represents varscan software indel detection result; *-o* means input conversion result file name

```
python $map_dir/bash/mutectTosclust.py -i $varscan2.out.indel.vcf \
                                       -o ${sclust}.input.vcf
```

The function of the script named *sclust_to_pyclone.py* is to convert sclust output file to Pyclone input file. *-i* represents allele information file, which can be obtained by sclust; *-n* means a sample name to control the output file name; -v the vcf file for sclust, and the previous The output files of the two scripts are the same.

```
python $map_dir/bash/sclust_to_pyclone.py -i ${sclust}.allelic.txt \
                                          -n $sample_name \
                                          -v ${sclust}.input.vcf \
                                          -o $output path of Directory
```

The function of the script named sclust_to_fastclone.py is to convert sclust output file to FastClone input file. Fastclone has the same input format as pyclone, so the

conversion script is the same.

```
python $map_dir/bash/ sclust_to_fastclone.py -i ${sclust}.allelic.txt \
                                             -n $sample_name \
                                             -v ${sclust}.input.vcf \
                                             -o $output path of Directory
```

The following is an example of a conversion file. Figure 4 (A) represents the output file of varscan, and Figure 4 (B) represents the input file of sclust after conversion.

```
chr1  206641  .  A  G  .  PASS  DP=39;SS=1;SSC=11;GPV=0.0058219;SPV=0.071064    GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:1
chr1  206680  .  T  C  .  PASS  DP=49;SS=1;SSC=11;GPV=0.013285;SPV=0.069782 GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:26:25
chr1  206693  .  T  G  .  PASS  DP=49;SS=1;SSC=12;GPV=0.0062092;SPV=0.055076    GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:2
chr1  206696  .  AG A  .  PASS  DP=50;SS=1;SSC=11;GPV=0.013331;SPV=0.078014 GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:26:25
chr1  206724  .  A  G  .  PASS  DP=51;SOMATIC;SS=2;SSC=17;GPV=1;SPV=0.016436    GT:GQ:DP:RD:AD FREQ:DP4 0/0:.:2
chr1  206733  .  A  AAG .  PASS  DP=57;SOMATIC;SS=2;SSC=25;GPV=1;SPV=0.0031178   GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:2
chr1  206808  .  A  G  .  PASS  DP=38;SS=1;SSC=10;GPV=0.012628;SPV=0.089856 GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:19:18
chr1  206817  .  G  T  .  PASS  DP=35;SS=1;SSC=6;GPV=0.026693;SPV=0.20674   GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:16:15
chr1  206849  .  C  G  .  PASS  DP=19;SS=1;SSC=4;GPV=0.051948;SPV=0.38235   GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:9:7:1
chr1  206863  .  T  C  .  PASS  DP=20;SS=1;SSC=4;GPV=0.11486;SPV=0.37564    GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:11:10
chr1  206946  .  T  C  .  PASS  DP=35;SS=1;SSC=6;GPV=0.01238;SPV=0.2478 GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:19:17:2:1
chr1  206949  .  A  C  .  PASS  DP=36;SS=1;SSC=6;GPV=0.012467;SPV=0.2265    GT:GQ:DP:RD:AD:FREQ:DP4 0/0:.:20:18
chr1  207010  .  T  C  .  PASS  DP=41;SS=1;SSC=3;GPV=9.5339e-05;SPV=0.4049  GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:20:15
chr1  207015  .  A  G  .  PASS  DP=42;SS=1;SSC=4;GPV=4.0777e-05;SPV=0.32338 GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:20:15
chr1  207108  .  T  A  .  PASS  DP=31;SS=1;SSC=1;GPV=0.0001594;SPV=0.71296  GT:GQ:DP:RD:AD:FREQ:DP4 0/1:.:13:8:
```

Figure 4 (A) The result file of varscan2.

```
chr1  28441   .  A   G   .  mfilter DP=13;AF=0.153846153846154;DP_N=14;AF_N=0
chr1  28875   .  G   A   .  mfilter DP=19;AF=0.421052631578947;DP_N=19;AF_N=0.0526315789473684
chr1  33520   .  G   GAT .  mfilter DP=18;AF=0.111111111111111;DP_N=25;AF_N=0
chr1  38323   .  C   T   .  mfilter DP=16;AF=0.125;DP_N=13;AF_N=0
chr1  39015   .  A   C   .  mfilter DP=18;AF=0.111111111111111;DP_N=11;AF_N=0
chr1  39256   .  A   G   .  PASS    DP=56;AF=0.232142857142857;DP_N=43;AF_N=0.0232558139534884
chr1  54001   .  A   G   .  PASS    DP=21;AF=0.142857142857143;DP_N=33;AF_N=0
chr1  84002   .  G   GAAAGAAAA   .  PASS    DP=18;AF=0.166666666666667;DP_N=39;AF_N=0
chr1  99081   .  T   C   .  PASS    DP=8;AF=0.25;DP_N=58;AF_N=0
chr1  113109  .  A   G   .  PASS    DP=34;AF=0.147058823529412;DP_N=45;AF_N=0
chr1  137114  .  A   T   .  mfilter DP=12;AF=0.166666666666667;DP_N=9;AF_N=0
chr1  142117  .  TG  T   .  PASS    DP=18;AF=0.222222222222222;DP_N=16;AF_N=0
chr1  159007  .  AAC A   .  mfilter DP=18;AF=0.111111111111111;DP_N=28;AF_N=0
chr1  162305  .  A   T   .  mfilter DP=15;AF=0.133333333333333;DP_N=17;AF_N=0
```

Figure 4 (B) The converted file for sclust.

Step-2 Use the *CN* command in Sclust

This step is responsible for calculating copy number variation (CNV), major and minor of copy number, clone copy number fraction (CCF), ploidy and purity in the WGS data.

```
Sclust cn -rc ${sampleID}_rcount.txt -snp ${sampleID}_snps.txt \
-vcf ${sclust}.input.vcf -o ${sampleID} --ns 1000
```

Step-3 Get VAF information from SNP file.

VAF can be found in the VCF format file, that is, the contents marked by the red box in Figure 4(a). The VAF value can be extracted by using the script in step 1

Step-4 Convert the format of input file.

The input formats of the 12 kinds of software are different. According to the different input formats, python scripts are used to format and output the data. Taking the detection results of mutect2 as an example, the document shows the output formats of 12 methods.

(a) Call pandas;

Pandas, which is a python extension package for processing big data.

```
import pandas as pd
```

(b) Use Python to read the detection results of mutect2;

The parameter path represents the path of the vcf file.

```
f = open(path, 'r')
df = pd.read_csv(f,sep='\t',header=None,comment='#')
df.columns =['CHROM','POS','ID','REF','ALT','QUAL','FILTER','INFO','FORMAT','TUMOR']
print("read end")
```

(c) *calculate the VAF value at each variant locus.*

Obtain the reference depth reading and allele reading depth to calculate the VAF value at each variant locus.

```
df_TUMOR = df['TUMOR'].str.split(":",expand=True)
df_TUMOR.columns=['GT','AD','AF','DP','F1R2','F2R1','PGT','PID','PS','SB']
df_TUMOR['DP'] = df_TUMOR['DP'].astype(int)
df_TUMOR['RD'] = df_TUMOR['AD'].str.split(",").str[0].astype(int)
df_TUMOR['AD'] = df_TUMOR['AD'].str.split(",").str[1].astype(int)
answer['DP_c'] = df_TUMOR['RD']+df_TUMOR['AD']
answer['AF_c'] = df_TUMOR['AD']/answer['DP_c']
df = df.join(answer)
```

(d) calculate the genotype of each variant locus

allelic_ File is the path of *${SampleID}_icn.seg* which is obtained by using the CN command of sclust software

```
f = open(allelic_file, 'r')
df2 = pd.read_csv(f, sep='\t', header=0, comment='#')
df2['Start'] = df2['Start'].astype(int)
df2['End'] = df2['End'].astype(int)
for chr,pos in zip(df[['CHROM','POS']]):
    aline = df2[(df2['Start'] < pos) & (df2['End'] > pos) & (df2['Chromosome'] == chr)]
    aline = aline[['Chromosome', 'A', 'B']]
    aline = aline.reset_index()
    if aline.shape[0] == 0:
        major = minor = 1
    else:
        major = aline['A'][0]
        minor = aline['B'][0]
```

(e) Filter

The genes located in 23 pairs of chromosomes, high reading depth and somatic variation were retained in the data

```
CHR = ['chr1','chr2','chr3','chr4','chr5','chr6','chr7','chr8','chr9','chr10','chr11','chr12',
'chr13','chr14','chr15','chr16','chr17','chr18','chr19','chr20','chr21','chr22','chrX','chrY']
df = df[df['CHROM'].isin(CHR)]
df = df[df['FILTER']=='PASS']
message = df[(df['DP_c']>14) & (df['AF_c']>0.1) ]
```

```
message = message[['CHROM','POS','REF','ALT','DP_n','AF_n','AD_n','DP_c','AF_c','AD_c','major'
,'minor','Sample']]
```

(f) Output files according to different formats

Sclust:

Sclust input has been processed in Step-1

PyClone and FastClone:

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write(message['Sample'].str+message['CHROM'].str+':'+message['POS'].str + '\t')
      file_out.write(message['DP_c'] + '\t')
      file_out.write(message['AD_c'] + '\t')
      file_out.write(message['major']  + '\t')
      file_out.write(message['minor'] + '\t')
      file_out.write(message['Sample']+ '\t')
      file_out.write(AB + '\t' + '\n')
```

DPClust:

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write(message['CHROM'].str + '\t')
      file_out.write(message['POS'].str  + '\t')
      file_out.write(message['DP_c'].str + '\t')
      file_out.write(message['AD_c'].str + '\t')
      file_out.write(int(message['major']+message['minor'])  + '\t')
      file_out.write(int(message['major']+message['minor'])  + '\t')
      file_out.write(message['AF_c']+ '\t')
      file_out.write(unphased + '\t' + '\n')
```

phyloWGs:

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write('s'+message['CHROM'].index + '\t')
      file_out.write(message['Sample'].str+message['CHROM'].str+':'+message['POS'].str + '\t')
      file_out.write(message['DP_n'].str + '\t')
      file_out.write(message['DP_c'].str + '\t')
      file_out.write(message['AF_n'].str  + '\t')
      file_out.write(message['AF_c'].str  + '\t'+ '\n')
```

sciClone and CLiP:

```
with open(output_dir_cnv + sample + '.tsv', 'w') as file_out:
      file_out.write(df2['CHROM'].str+ '\t')
      file_out.write(df2['Start'].str + '\t')
      file_out.write(df2['End'].str + '\t'+ '\n')
      file_out.write(int(df2['major']+ df2['minor']) + '\t')
with open(output_dir_snp + sample + '.tsv', 'w') as file_out:
      file_out.write(message['CHROM'].str + '\t')
      file_out.write(message['POS'].str + '\t')
      file_out.write(message['DP_n'].str + '\t')
      file_out.write(message['DP_c'].str + '\t')
      file_out.write(message['AF_c'].str + '\t'+ '\n')
```

Svclone

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write(message['CHROM'].str + '\t')
      file_out.write(message['POS'].str + '\t')
      file_out.write(message['CHROM'].str + '\t')
      file_out.write(message['POS'].str + '\t')
```

TrAp

```
with open(output_dir + sample + '.html', 'w') as file_out:
      file_out.write('SIGNAL' + '\t')
      file_out.write(message['Sample'].str+message['CHROM'].str+':'+message['POS'].str + ' ')
      file_out.write(message['DP_c'].str + ' '+ '\n')
```

CloneFinder

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write(message['Sample'].str+message['CHROM'].str+':'+message['POS'].str + '\t')
      file_out.write(message['DP_n'].str + '\t')
      file_out.write(message['DP_c'].str + '\t'+ '\n')
```

PhylogicNDT:

```
with open(output_dir + sample + '.tsv', 'w') as file_out:
      file_out.write(message['Sample'].str+message['CHROM'].str+':'+message['POS'].str + '\t')
      file_out.write(message['CHROM'].str + '\t')
      file_out.write(message['POS'].str + '\t')
      file_out.write(message['REF'].str + '\t')
      file_out.write(message['ALT'].str + '\t')
      file_out.write(message['DP_c'].str  + '\t')
      file_out.write(message['AD_c'].str  + '\t'+ '\n')
```

4. Subclonal inferencing

Sclust

1 bamprocess

The function of *bamprocess* is to analyze the reading depth (RD) information, allele information and GC content and SNP of whole genome sequencing (WGS) data. This information is stored in two files named ${sampleID}_rcount.txt and ${sampleID}_snps.txt. In particular, each chromosome needs to be processed separately. Finally, the *-i* and *-o* parameters are used to merge all the files to get the final result.

```
Sclust bamprocess -t ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -n ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -o ${sampleID} -part 2 -build hg38 -r chr1

Sclust bamprocess -t ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -n ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -o ${sampleID} -part 2 -build hg38 -r chr2
... ...
```

```
Sclust bamprocess -t ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -n ${sampleID}.WGS.sorted.markdup.bqsr.bam \
                  -o ${sampleID} -part 2 -build hg38 -r chrY
Sclust bamprocess -i ${sampleID}. -o ${sampleID}.
```

2 cn

This step is responsible for calculating copy number variation (CNV), clone copy number fraction (CCF), ploidy and purity in the WGS data.

```
Sclust cn -rc ${sampleID}_rcount.txt -snp ${sampleID}_snps.txt \
-vcf ${sclust}.input.vcf -o ${sampleID} --ns 1000
```

3 cluster

This function is used to cluster the tumor cell fraction (CCF) to find the corresponding evolution table.

```
Sclust cluster -I ${sampleID} -lambda ${lambda}
```

Figure 4 shows some visualization results of sclust software, in which Figure 4 (A) shows the purity, ploidy and copy number of tumor, and Figure 4 (B) describes the clustering result information.

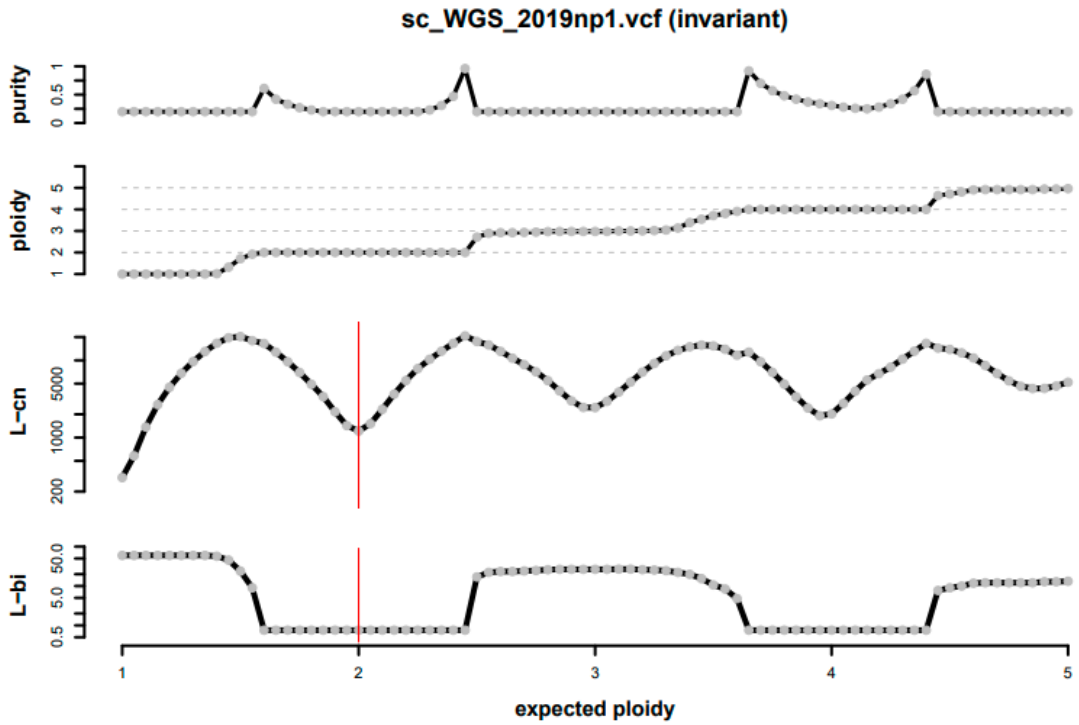

Figure 4 (A) The result files obtained by running cn function.

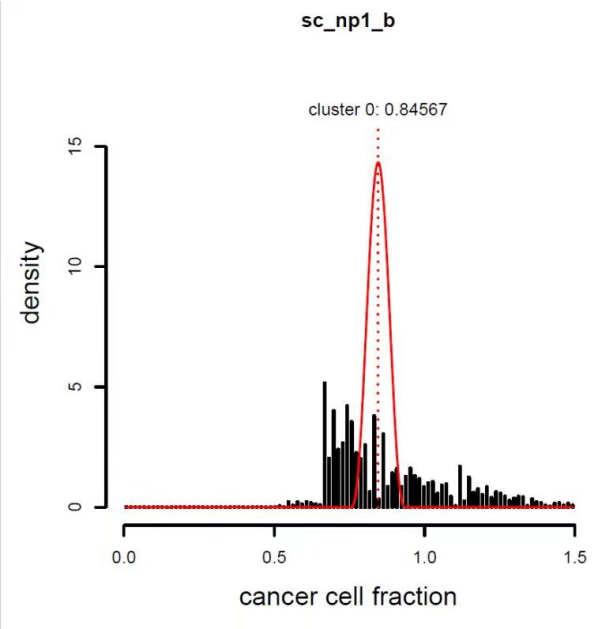

Figure 4 (B) The result files obtained by running cluster function.

Pyclone

1 analysis pipeline

Run the whole process of pyclone to get the phylogenetic tree of tumor cells.

2 setup and run analysis

Pyclone uses the function named *setup_analysis* to specify the location of the input sample and the output of the result file, and generates the configuration file in yaml format.

```
PyClone setup_analysis --in_files ${sampleID}/*.tsv --working_dir ${sampleID}_out_dir
```

The function of *run_analysis* specifies the output location of yaml format configuration files, and the command will output yaml format files.

```
PyClone run_analysis --config_file ${sampleID}_out_dir/config.yaml
```

3 plot clusters

The function named *plot_clusters* generates the clustered images, outputs them in PDF format, and specifies the location of the PDF file.

```
PyClone plot_clusters --config_file ${sampleID}_out_dir/config.yaml \
--plot_file ${sampleID}_out_dir/cluster --plot_type density
```

4 plot loci

The function of *plot_clusters* generates the image file of loci and outputs it in PDF format

```
PyClone plot_loci --config_file ${sampleID}_out_dir/config.yaml \
--plot_file ${sampleID}_out_dir/cluster --plot_type density
```

Figure 5 shows part of the results of the plot clusters function, which showed the relationship between cell prevalence and clustering density.

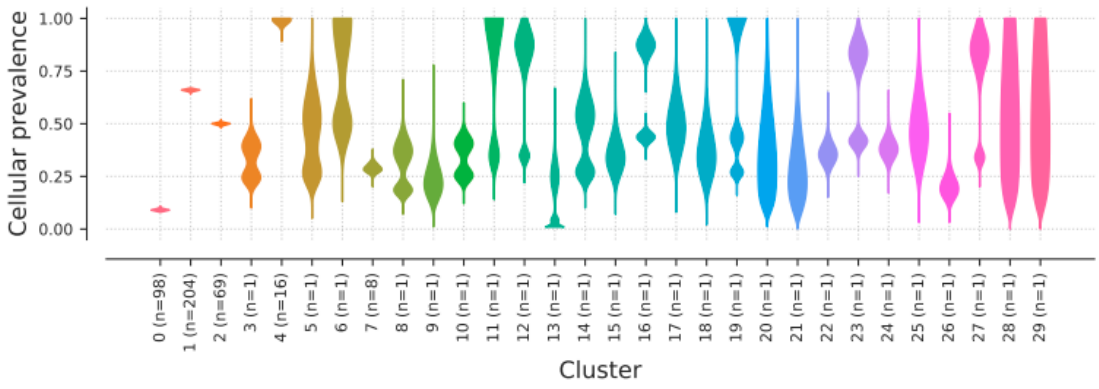

Figure 5 The visualization result of pyclone

Fastclone

1 run fastclone

fastclone contains three key parameters, load-pyclone, load-pyclone_truth and load-mutect_battenberg. Using the load-pyclone parameter means that the loaded file is the same as the input file of pyclone. The load-pyclone_truth parameter represents that the loaded file is selected from the pyclone input file. load-mutect_battenberg represents that the loaded file is the file obtained by the MuTect and Battenberg methods. The specific running code is as follows:

```
for sample_file in ${sampleID}_dir
do
(fastclone load-pyclone prop ${sample_file} None solve \
${sample_file}_output) >> output_record_update 2>&1
done
```

5. The formats of all 12 methods

sclust:

Sclust software needs to use SNP results of sequencing data in *CN* command. The input format of SNP information is VCF, and the details are as follows:

Column	Description
CHROM	Chromosome on which the mutation occurred
POS	Position at which the mutation occurred
ID	Any string identifying the variant -- this need not be a gene name.
REF	Base type of reference gene
ALT	Base type allele gene
Filter	Base quality
INFO	Mutation information

PyClone and FastClone:

Pyclone and fastclone have the same input format. The specific format is as follows:

Column	Description
mutation_id	Any string identifying the variant -- this need not be a gene name.
ref_counts	The number of reads overlapping the locus matching the reference allele
var_counts	The number of reads overlapping the locus matching the variant allele.
normal_cn	The copy number of the locus in non-malignant cells. This should generally be 2 except for sex chromosomes in males.
minor_cn	The copy number of the minor allele in the malignant cells. This must be less than equal the value in the major_cn column.
major_cn	The copy number of the major allele in the malignant cells. This should be greater than equal to the value in the minor_cn column and greater than 0.

DPClust:

The detailed meaning of each column in the input file is as follows:

Column	Description
chr	Chromosome on which the mutation occurred
end	Position at which the mutation occurred
WT.count	The number of sequencing reads supporting the reference allele
mut.count	The number of sequencing reads supporting the mutation allele
subclonal.CN	The total copy number at the location of the mutation
mutation.copy.number	The raw estimate of the average number of chromosome copies that carry the mutation
subclonal.fraction	The estimate of the fraction of tumour cells (CCF) that carry the mutation
no.chrs.bearing.mut	The mutation's multiplicity estimate

phyloWGs:

The detailed meaning of each column in the input file is as follows:

Column	Description
id	Identifier for each SSM. Identifiers must start at s0 and increment, so the first data row will have s0, the second row s1, and so forth.
gene	Any string identifying the variant -- this need not be a gene name.
a	Number of reference-allele reads at the variant locus.
d	Total number of reads at the variant locus.
mu_r	Fraction of expected reference allele sampling from the **reference** population
mu_v	Fraction of expected reference allele sampling from **variant** population

sciClone:

Sciclone has two input files that record CNV and SNP information respectively. The file format for recording **CNV** information is as follows:

Column	Description
chr	Chromosome on which the mutation occurred
start	Starting position of copy number variation segment
stop	Endding position of copy number variation segment

Column	Description
segment_mean	The absolute copy number of the segment

The file format for recording **SNP** information is as follows:

Column	Description
chr	Chromosome on which the mutation occurred
pos	The position of SNP
ref_reads	Read depth of reference-allele reads at the variant locus.
var_reads	Read depth of variant-allele reads at the variant locus.
vaf	Variant Allele Frequency

Svclone

Column	Description
Chr1	Chromosome in which structural variation occurs
Pos1	Location of structural variation
Chr2	Chromosome in which structural variation occurs
Pos2	Location of structural variation

CLIP

CLiP has three input files that record CNV, SNP and purity information respectively. The file format for recording **CNV** information is as follows:

Column	Description
chr	Chromosome on which the mutation occurred
start	the start position of the CNA segment on the corresponding chromosome.
end	the end position of the CNA segment on the corresponding chromosome
major_cn	The copy number of the major allele in tumor cells. This should be greater than equal to the value in the minor_cn column and greater than 0
minor_cn	The copy number of the minor allele. This must be less than equal the value in the major_cn column.
total_cn	The sum of major_cn and minor_cn.

The file format for recording **SNP** information is as follows:

Column	Description
chromosome_index	The chromosomal location of the SNV.

Column	Description
position	the single-nucleotide position of the SNV on the corresponding chromosome.
ref_count	The number of reads covering the locus and containing the reference allele
alt_count	The number of reads covering the locus and containing the alternative allele

The file format for recording P**urity** information is as follows:

Column	Description
Number	Decimal representing tumor purity

TrAp

Column	Description
name	unique identifier of the genomic aberration
value	cellular frequency of the genomic aberration

CloneFinder

Column	Description
"XX:ref"	Reference read count for the sample, XX
"XX:alt"	Variant read count for the sample, XX

PhylogicNDT

Column	Description
Hugo_Symbol	Gene name
Chromosome	The chromosomal location of the Variant
Start_position	the start position of the SNP segment on the corresponding chromosome.
Reference_Allele	Base type of reference gene
Tumor_Seq_Allele2	Base type after mutation
t_ref_count	The number of reads covering the locus and containing the reference allele
t_alt_count	The number of reads covering the locus and containing the alternative allele