Input 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:

```
##fileformat=VCFv4.1
##source=ChangefromVarScan2
##INFO=<ID=DP,Number=1,Type=Integer, Description="Read Depth Tumor">
##INFO=<ID=DP_N,Number=1,Type=Integer, Description="Read Depth Normal">
##INFO=<ID=AF,Number=A,Type=Float, Description="Allelic Frequency Tumor">
##INFO=<ID=AF_N,Number=A,Type=Float, Description="Allelic Frequency Normal">
##INFO=<ID=FR,Number=1,Type=Float, Description="Forward-Reverse Score">
##INFO=<ID=TG,Number=1,Type=String, Description="Target Name (Genome Partition)">
##INFO=<ID=DB,Number=0,Type=Flag, Description="dbSNP Membership">
##mfilterParameters= -af 0.2 -fr 0.2 -rc 5
#CHROM POS      ID      REF     ALT     QUA     FILTER  INFO
chr11   194441  .       C       T       .       PASS    DP=15;AF=0.2;DP_N=15;AF_N=0
chr11   18744169       .       C       CA      .       PASS    DP=16;AF=1.0;DP_N=16;AF_N=0
chr11   18751041       .       C       T       .       PASS    DP=19;AF=0.157894736842;DP_N=19;AF_N=0
chr11   18752633       .       C       T       .       PASS    DP=28;AF=0.964285714286;DP_N=28;AF_N=0
chr11   18757355       .       G       A       .       PASS    DP=25;AF=0.96;DP_N=25;AF_N=0
chr11   18762983       .       T       G       .       PASS    DP=38;AF=0.947368421053;DP_N=38;AF_N=0
chr11   18772655       .       C       T       .       PASS    DP=21;AF=0.809523809524;DP_N=21;AF_N=0
chr11   18776331       .       T       C       .       PASS    DP=27;AF=0.851851851852;DP_N=27;AF_N=0
chr11   18776461       .       TC      T       .       PASS    DP=25;AF=1.0;DP_N=25;AF_N=0
chr11   18783768       .       T       C       .       PASS    DP=29;AF=1.0;DP_N=29;AF_N=0
chr11   18814983       .       T       G       .       PASS    DP=24;AF=0.916666666667;DP_N=24;AF_N=0
chr11   20426090       .       G       A       .       PASS    DP=15;AF=1.0;DP_N=15;AF_N=0
```

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:

chr	end	WT.count	mut.count	subclonal.CN	mutation.copy.number	subclonal.fraction	no.chrs.bearing.mut	phase
1	1	38	42	2	1.05	1.05	1	unphased
1	2	44	55	2	1.11111111111111	1.11111111111111	1	unphased
1	3	50	48	2	0.979591836734694	0.979591836734694	1	unphased
1	4	51	52	2	1.00970873786408	1.00970873786408	1	unphased
1	5	40	46	2	1.06976744186047	1.06976744186047	1	unphased
1	6	69	45	2	0.789473684210526	0.789473684210526	1	unphased
1	7	42	64	2	1.20754716981132	1.20754716981132	1	unphased
1	8	42	57	2	1.15151515151515	1.15151515151515	1	unphased
1	9	53	48	2	0.95049504950495	0.95049504950495	1	unphased
1	10	56	47	2	0.912621359223301	0.912621359223301	1	unphased
1	11	36	56	2	1.21739130434783	1.21739130434783	1	unphased
1	12	50	44	2	0.936170212765957	0.936170212765957	1	unphased
1	13	45	38	2	0.91566265060241	0.91566265060241	1	unphased
1	14	44	64	2	1.18518518518519	1.18518518518519	1	unphased
1	15	54	53	2	0.990654205607477	0.990654205607477	1	unphased
1	16	57	50	2	0.934579439252336	0.934579439252336	1	unphased
1	17	43	46	2	1.03370786516854	1.03370786516854	1	unphased
1	18	43	47	2	1.04444444444444	1.04444444444444	1	unphased
1	19	54	46	2	0.92	0.92	1	unphased

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:

id	gene	a	d	mu_r	mu_v
s0	S0	71	106	0.999	0.5
s1	S1	83	112	0.999	0.5
s2	S2	66	91	0.999	0.5
s3	S3	72	90	0.999	0.5
s4	S4	67	87	0.999	0.5
s5	S5	80	100	0.999	0.5
s6	S6	78	99	0.999	0.5
s7	S7	93	113	0.999	0.5
s8	S8	65	88	0.999	0.5
s9	S9	83	103	0.999	0.5
s10	S10	77	105	0.999	0.5
s11	S11	87	106	0.999	0.5
s12	S12	85	112	0.999	0.5

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
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

Column	Description
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.
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

```
DATATYPE FIXED 0.0000001
SIGNAL WT 1.
SIGNAL A<sub>2</sub> .6
SIGNAL A<sub>3</sub> .4
SIGNAL A<sub>4</sub> .35
SIGNAL A<sub>5</sub> .3
SIGNAL A<sub>6</sub> .1
```

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

CloneFinder

SNVID	Wild	Mut	R2:ref	R2:alt	R3:ref	R3:alt	R4:ref	R4:alt	R6:ref	R6:alt	R1:ref	R1:alt	R5:ref	R5:alt	R7:ref	R7:alt
S1	T	A	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S2	C	A	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S3	A	T	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S4	A	T	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S5	C	T	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S6	T	C	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S7	A	G	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S8	C	T	70	30	70	30	50	50	50	50	70	30	70	30	60	40
S9	T	G	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S10	A	G	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S11	A	C	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S12	T	A	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S13	C	G	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S14	C	T	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S15	A	G	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S16	C	A	100	0	100	0	100	0	70	30	100	0	100	0	100	0
S17	C	T	70	30	70	30	50	50	95	5	70	30	70	30	60	40
S18	T	G	70	30	70	30	50	50	95	5	70	30	70	30	60	40
S19	G	A	70	30	70	30	50	50	95	5	70	30	70	30	60	40

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

Column	Description
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

Convert mutect output file to sclust input file

For single sample

```
import warnings
import pandas as pd
import sys, getopt
warnings.simplefilter("ignore")
def main(argv):
	inputfile = ''
	outputfile = ''
	try:
		opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
	except getopt.GetoptError:
		print('test.py -i <inputfile> -o <outputfile>')
		sys.exit(2)
	for opt, arg in opts:
		if opt == '-h':
			print('test.py -i <inputfile> -o <outputfile>')
			sys.exit()
		elif opt in ("-i", "--ifile"):
			inputfile = arg
		elif opt in ("-o", "--ofile"):
			outputfile = arg
	return inputfile,outputfile

def Write_File(LIST,filename):
	output = open(filename, "w")
	for i in range(len(LIST)):
		output.write(LIST[i] + '\n')

def Add_File(LIST,filename):
	#output = open(filename, "a")
	LIST.to_csv(filename,mode='a',index=0,header=0,sep='\t')

def getfile_To_sclust(path):
	f = open(path, 'r')
	df = pd.read_csv(f,sep='\t',header=None,comment='#')
```

```
    print("read end")
    df.columns=['CHROM','POS','ID','REF','ALT','QUAL','FILTER','INFO','FORMAT','TUMOR']
    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']
    print("total list :" + str(df.shape[0]))
    df = df[df['CHROM'].isin(CHR)]
    df = df[df['FILTER']=='PASS']
    print("total list :" + str(df.shape[0]))
    #df_NOMAL= df['NORMAL'].str.split(":",expand=True)
    df_TUMOR = df['TUMOR'].str.split(":",expand=True)
    # Format: GT:AD:AF:DP:F1R2:F2R1:SB
    #df_NOMAL.columns=['GT','AD','AF','DP','F1R2','F2R1','SB']
    print(df_TUMOR)
    df_TUMOR.columns=['GT','AD','AF','DP','F1R2','F2R1','PGT','PID','PS','SB']
    df_TUMOR['VAF'] = df_TUMOR['AF']
    #df_NOMAL['VAF'] = df_NOMAL['AF']
    df = df[['CHROM','POS','ID','REF','ALT','QUAL','FILTER']]
    df_TUMOR['DP'] = df_TUMOR['DP'].astype(int)
    df_TUMOR['RD'] = df_TUMOR['AD'].str.split(",").str[0]
    df_TUMOR['AD'] = df_TUMOR['AD'].str.split(",").str[1]
    df_TUMOR['RD'] = df_TUMOR['RD'].astype(int)
    df_TUMOR['AD'] = df_TUMOR['AD'].astype(int)
    #df_NOMAL['DP'] = df_NOMAL['DP'].astype(int)
    #df_NOMAL['RD'] = df_NOMAL['AD'].str.split(",").str[0]
    #df_NOMAL['AD'] = df_NOMAL['AD'].str.split(",").str[1]
    answer = pd.DataFrame()
    answer['DP_c'] = df_TUMOR['RD']+df_TUMOR['AD']
    answer['AF_c'] = df_TUMOR['AD']/answer['DP_c']
    answer['DP_n'] = df_TUMOR['RD']+df_TUMOR['AD']
    answer['AF_n'] = 0     #设置为 2%
    #answer['DP_n'] = df_NOMAL['RD']+df_NOMAL['AD']
    #answer['AF_n'] = df_NOMAL['AD']/answer['DP_n']
    df = df.join(answer)
    message = df[(df['DP_c']>14) & (df['AF_c']>0.1) ]
    message[['DP_c', 'AF_c', 'DP_n', 'AF_n']] = message[['DP_c', 'AF_c', 'DP_n',
'AF_n']].astype(str)
    message['INFO'] = str('DP=') + message['DP_c'] + str(';AF=') + message['AF_c'] +
str(';DP_N=') + message[
        'DP_n'] + str(';AF_N=') + message['AF_n']
    message = message[['CHROM', 'POS', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO']]
    return message
def head():
    head = ['##fileformat=VCFv4.1',
```

```
                '##source=ChangefromVarScan2',
                '##INFO=<ID=DP,Number=1,Type=Integer, Description="Read Depth Tumor">',
                '##INFO=<ID=DP_N,Number=1,Type=Integer, Description="Read Depth
Normal">',
                '##INFO=<ID=AF,Number=A,Type=Float, Description="Allelic Frequency
Tumor">',
                '##INFO=<ID=AF_N,Number=A,Type=Float, Description="Allelic Frequency
Normal">',
                '##INFO=<ID=FR,Number=1,Type=Float, Description="Forward-Reverse
Score">',
                '##INFO=<ID=TG,Number=1,Type=String, Description="Target Name (Genome
Partition)">',
                '##INFO=<ID=DB,Number=0,Type=Flag, Description="dbSNP Membership">',
                '##mfilterParameters= -af 0.2 -fr 0.2 -rc 5',
                '#CHROM\tPOS\tID\tREF\tALT\tQUA\tFILTER\tINFO']
        return head
inputfile,outputfile=main(sys.argv[1:])
message = getfile_To_sclust(inputfile)
#print(message)
Write_File(head(),outputfile)
Add_File(message,outputfile)
```

For Comparison sample and variation sample

```
import warnings
import pandas as pd
import sys, getopt
warnings.simplefilter("ignore")
def main(argv):
        inputfile = ''
        outputfile = ''
        try:
                opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
        except getopt.GetoptError:
                print('test.py -i <inputfile> -o <outputfile>')
                sys.exit(2)
        for opt, arg in opts:
                if opt == '-h':
                        print('test.py -i <inputfile> -o <outputfile>')
                        sys.exit()
                elif opt in ("-i", "--ifile"):
                        inputfile = arg
                elif opt in ("-o", "--ofile"):
                        outputfile = arg
        return inputfile,outputfile
```

```
def Write_File(LIST,filename):
	output = open(filename, "w")
	for i in range(len(LIST)):
		output.write(LIST[i] + '\n')

def Add_File(LIST,filename):
	#output = open(filename, "a")
	LIST.to_csv(filename,mode='a',index=0,header=0,sep='\t')

def getfile_To_sclust(path):
	f = open(path, 'r')
	df = pd.read_csv(f,sep='\t',header=None,comment='#')
	print("read end")
	df.columns=['CHROM','POS','ID',
'REF','ALT','QUAL','FILTER','INFO','FORMAT','TUMOR','NORMAL']
	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']
	print("total list :" + str(df.shape[0]))
	df = df[df['CHROM'].isin(CHR)]
	df = df[df['FILTER']=='PASS']
	print("total list :" + str(df.shape[0]))
	df_NOMAL= df['NORMAL'].str.split(":",expand=True)
	df_TUMOR = df['TUMOR'].str.split(":",expand=True)
	# Format: GT:AD:AF:DP:F1R2:F2R1:SB
	df_NOMAL.columns=['GT','AD','AF','DP','F1R2','F2R1','PGT','PID','PS','SB']
	df_TUMOR.columns=['GT','AD','AF','DP','F1R2','F2R1','PGT','PID','PS','SB']
	df_TUMOR['VAF'] = df_TUMOR['AF']
	df_NOMAL['VAF'] = df_NOMAL['AF']
	df = df[['CHROM','POS','ID','REF','ALT','QUAL','FILTER']]
	df_TUMOR['DP'] = df_TUMOR['DP'].astype(int)
	df_TUMOR['RD'] = df_TUMOR['AD'].str.split(",").str[0]
	df_TUMOR['AD'] = df_TUMOR['AD'].str.split(",").str[1]
	df_TUMOR['RD'] = df_TUMOR['RD'].astype(int)
	df_TUMOR['AD'] = df_TUMOR['AD'].astype(int)
	df_NOMAL['DP'] = df_NOMAL['DP'].astype(int)
	df_NOMAL['RD'] = df_NOMAL['AD'].str.split(",").str[0]
	df_NOMAL['AD'] = df_NOMAL['AD'].str.split(",").str[1]
	df_NOMAL['RD'] = df_NOMAL['RD'].astype(int)
	df_NOMAL['AD'] = df_NOMAL['AD'].astype(int)
	answer = pd.DataFrame()
	answer['DP_c'] = df_TUMOR['RD']+df_TUMOR['AD']
	answer['AF_c'] = df_TUMOR['AD']/answer['DP_c']
	answer['DP_n'] = df_NOMAL['RD']+df_NOMAL['AD']
```

```
    answer['AF_n'] = df_NOMAL['AD']/answer['DP_n']
    df = df.join(answer)
    message = df[(df['DP_c']>14) & (df['AF_c']>0.1) ]
    message[['DP_c', 'AF_c', 'DP_n', 'AF_n']] = message[['DP_c', 'AF_c', 'DP_n', 'AF_n']].astype(str)
    message['INFO'] = str('DP=') + message['DP_c'] + str(';AF=') + message['AF_c'] + str(';DP_N=') + message['DP_n'] + str(';AF_N=') + message['AF_n']
    message = message[['CHROM', 'POS', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO']]
    return message
def head():
    head = ['##fileformat=VCFv4.1',
        '##source=ChangefromVarScan2',
        '##INFO=<ID=DP,Number=1,Type=Integer, Description="Read Depth Tumor">',
        '##INFO=<ID=DP_N,Number=1,Type=Integer, Description="Read Depth Normal">',
        '##INFO=<ID=AF,Number=A,Type=Float, Description="Allelic Frequency Tumor">',
        '##INFO=<ID=AF_N,Number=A,Type=Float, Description="Allelic Frequency Normal">',
        '##INFO=<ID=FR,Number=1,Type=Float, Description="Forward-Reverse Score">',
        '##INFO=<ID=TG,Number=1,Type=String, Description="Target Name (Genome Partition)">',
        '##INFO=<ID=DB,Number=0,Type=Flag, Description="dbSNP Membership">',
        '##mfilterParameters= -af 0.2 -fr 0.2 -rc 5',
        '#CHROM\tPOS\tID\tREF\tALT\tQUA\tFILTER\tINFO']
    return head
inputfile,outputfile=main(sys.argv[1:])
message = getfile_To_sclust(inputfile)
Write_File(head(),outputfile)
Add_File(message,outputfile)
```