

Zhang-Lab 生信小课堂 第14期

和趣求真  秉实生信

(张建伟生物信息学课题组 <https://zhang.hzau.edu.cn>)

序列比对SAM文件格式及操作

The Sequence Alignment/Map format and operation

地点：二综一楼C102 时间：15:00 (2023.09.08)

欢迎大家一起交流学习!

主讲人：赵志远

2023/09/08

Zhang-Lab 生信小课堂 第14期

和趣求真  秉实生信

(张建伟生物信息学课题组 <https://zhang.hzau.edu.cn>)

序列比对SAM文件格式及操作

The Sequence Alignment/Map format and operation

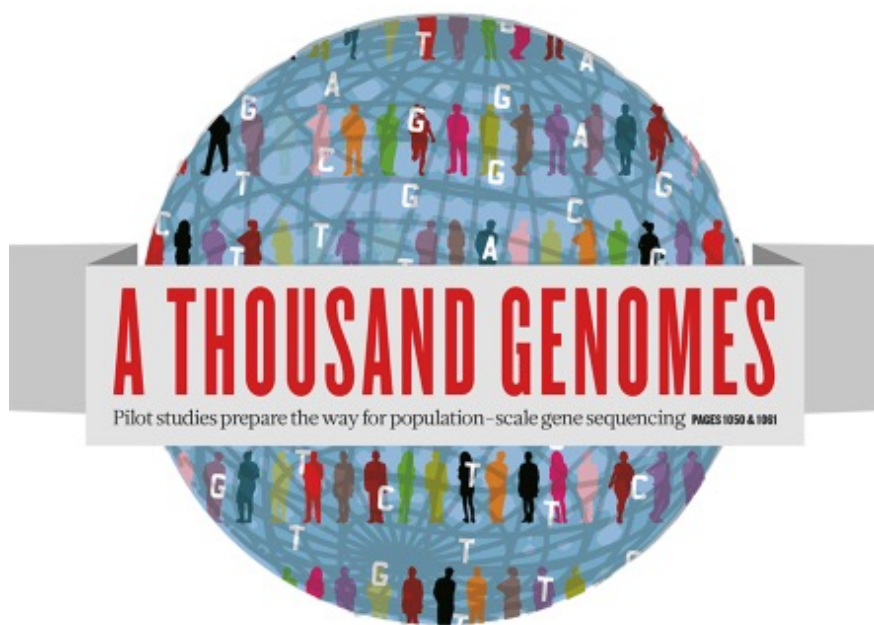
主讲人：赵志远

2023/09/08

千人基因组计划

nature

THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE



人类基因组计划(HGP)的完成获得的人类参考基因组序列, 为人类遗传学研究提供了基础, 但深入了解人类遗传多样性需要全面了解全部等位基因频率和DNA序列变异。

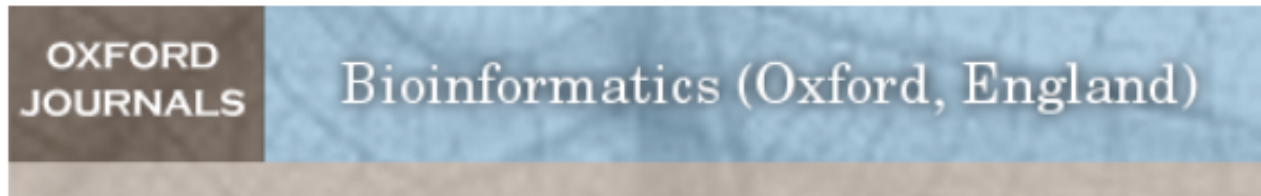
为了构建全面、详细的人类遗传变异图谱, **千人基因组计划(1000 Genomes Project, 1KGP)**于2008年1月启动, 该计划的目标是在多个不同的人群中找到频率至少为1%的常见遗传变异。

1KGP项目成果

- 多个人群数千个体的基因组测序
- 人类基因型和遗传变异的鉴定
- 全球人类遗传多样性的研究
- 人类疾病关联基因的发现
- 数据库的建立与分析工具的开发



SAM文件格式



[Bioinformatics](#). 2009 Aug 15; 25(16): 2078–2079.

PMCID: PMC2723002

Published online 2009 Jun 8. doi: [10.1093/bioinformatics/btp352](https://doi.org/10.1093/bioinformatics/btp352)

PMID: [19505943](https://pubmed.ncbi.nlm.nih.gov/19505943/)

The Sequence Alignment/Map format and SAMtools

[Heng Li](#),^{1,†} [Bob Handsaker](#),^{2,†} [Alec Wysoker](#),² [Tim Fennell](#),² [Jue Ruan](#),³ [Nils Homer](#),⁴ [Gabor Marth](#),⁵ [Goncalo Abecasis](#),⁶ [Richard Durbin](#),^{1,*} and 1000 Genome Project Data Processing Subgroup⁷

序列比对/映射(Sequence Alignment/Map, SAM)格式是一种通用比对格式，也是千人基因组计划发布比对的格式，用于存储针对参考序列的读段比对，支持不同测序平台产生的短读段和长读段。SAM格式风格灵活、尺寸紧凑、随机访问高效，配合比对后处理工具，如SAMtools等可以高效处理序列比对结果。



SAM文件格式

在SAM(v1.4)中，每条对齐线都有11个必填字段和数量可变的可选字段。

```

Coord      12345678901234  5678901234567890123456789012345
ref        AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1    TTAGATAAAGGATA*CTG
+r002      aaaAGATAA*GGATA
+r003      gcctaAGCTAA
+r004      ATAGCT.....TCAGC
-r003      ttagctTAGGC
-r001/2    CAGCGGCAT
    
```

```

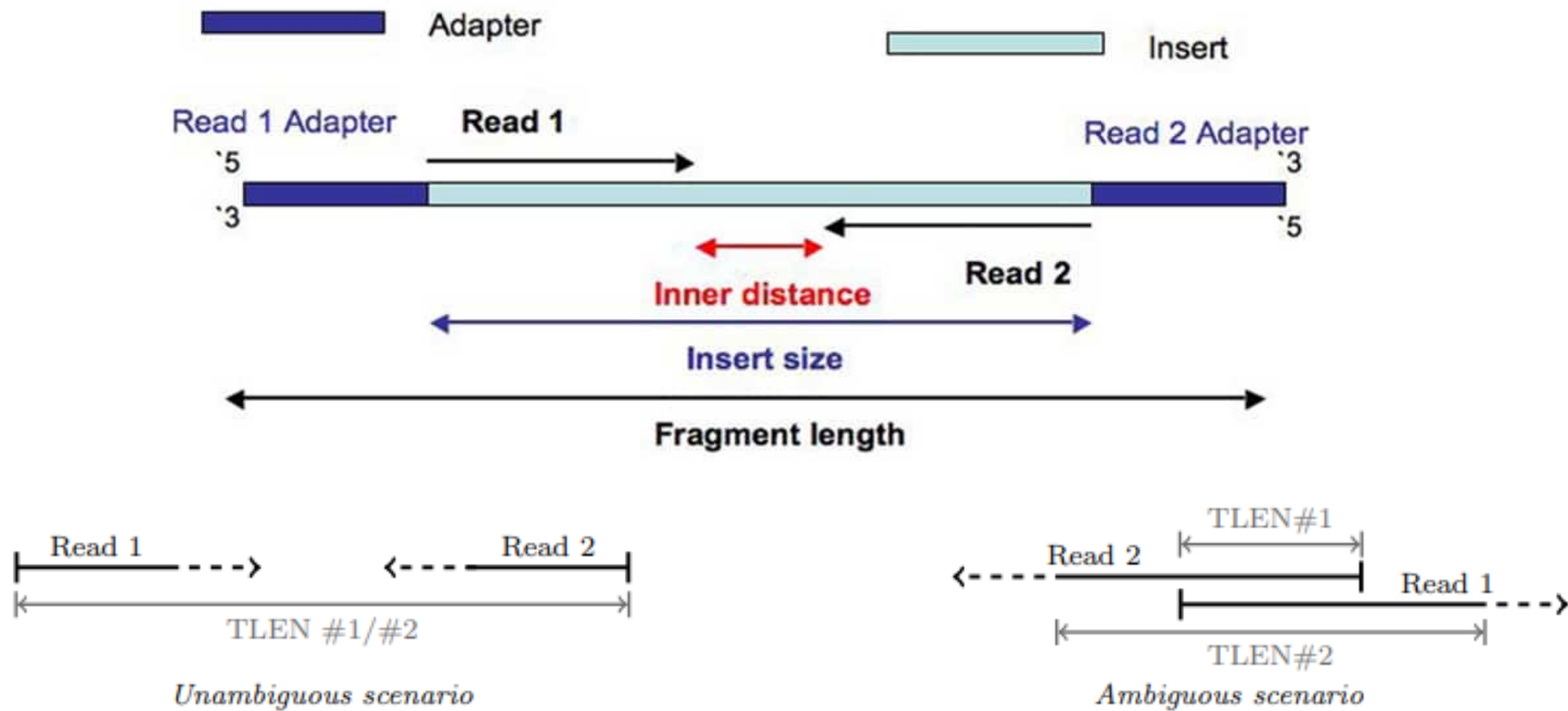
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA *
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC *
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT *
    
```

No.	名称	描述
1	QNAME	查询(Query)序列名称
2	FLAG	比对信息位
3	RNAME	参考(REF)序列名称
4	POS	比对位置
5	MAPQ	比对质量值
6	CIGAR	CIGAR 字符串
7	RNEXT	配对序列参考名称
8	PNEXT	配对序列比对位置
9	TLEN	模板序列长度
10	SEQ	匹配部分序列
11	QUAL	序列碱基质量



SAM文件格式

Paired-end测序



SAM文件格式

FLAG(C2, 比对信息位)

```

Coord      12345678901234  5678901234567890123456789012345
ref        AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1    TTAGATAAAGGATA*CTG
+r002      aaaAGATAA*GGATA
+r003      gcctaAGCTAA
+r004      ATAGCT.....TCAGC
-r003      ttagctTAGGC
-r001/2    CAGCGGCAT
    
```

```

@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001 99 =64+32+2+1 存在配对序列; 序列均正确匹配;
r002 0 =0 配对序列的反向互补匹配; Read1
r003 0 =0 正链匹配
r004 0 =0
r003 2064 =2048+16 嵌合匹配; 反向互补匹配
r001 147 =128+16+2+1 存在配对序列; 序列均正确匹配;
反向互补匹配; Read2
    
```

整数	描述
1	存在配对序列
2	该序列与配对序列均正确匹配
4	该序列未匹配
8	配对序列未匹配
16	该序列的反向互补序列匹配
32	配对序列的反向互补序列匹配
64	配对测序的第一条序列
128	配对测序的第二条序列
256	次级匹配结果
512	序列质量控制不合格
1024	PCR重复或光学重复
2048	嵌合匹配的部分序列



SAM文件格式

CIGAR(C6, 比对信息字符串)

```

Coord      12345678901234  5678901234567890123456789012345
ref        AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1    TTAGATAAAGGATA*CTG
+r002      aaaAGATAA*GGATA
+r003      gcctaAGCTAA
+r004      ATAGCT.....TCAGC
-r003      ttagctTAGGC
-r001/2    CAGCGGCAT
    
```

```

@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA *
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC *
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT *
    
```

字符	描述
M	匹配 (包含错配)
I	插入
D	缺失
N	跳过
S	软跳过(不改变序列长度)
H	硬跳过(改变序列长度)
P	填补
=	匹配
X	错配



SAM文件操作

SAMtools是一个库和软件包，用于解析和操作SAM格式的序列比对文件。

功能	命令	描述
查看	view	SAM <=> BAM <=> CRAM
	tview	交互式的查看reads比对信息
索引	index	对BAM文件建索引
	faidx	对fasta参考基因组建索引
文件操作	sort	排序
	merge	合并多个文件
	fasta	BAM => FASTA
	mpileup	堆积格式
统计	flagstat	比对信息统计
	depth	每个碱基测序深度统计



SAM文件操作

Pysam是一个 Python 模块, 可以轻松读取和操作SAM/BAM 文件

工具	SAMtools	Pysam
运行环境	Linux/Shell	Python
处理速度	快	稍慢
编程要求	低	略高 (需要对Python有一定了解)
代码量	少	较多
社区支持	良好	较差
灵活性	功能丰富但有限	高
自定义	限制较多	友好



SAM文件操作

Pysam 安装

- **Conda安装**

```
conda config --add channels r  
conda config --add channels bioconda  
conda install pysam
```

- **Pypi安装**

```
pip install pysam
```



SAM文件操作

文件读取

```
class AlignmentFile(filepath_or_object, mode=None, ...)
```

Parameters:

filepath_or_object: file path string or python File object

mode: r - reading; w - writing; b - bam; c - cram

```
import pysam
samfile = pysam.AlignmentFile("test.sam", "r")
samfile = pysam.AlignmentFile("test.bam", "rb")
```

```
for read in samfile.fetch():
    print(read)
```

```
>>> for read in samfile.fetch():
...     print(read)
...
r001  99      #0      7      30      8M2I4M1D3M      #0      37      39      TTAGATAAAGGATACTG      None      []
r002  0        #0      9      30      3S6M1P1I4M      *      0        0        AAAAGATAAGGATA      None      []
r003  0        #0      9      30      5S6M      *      0        0        GCCTAAGCTAA      None      [('SA', 'ref,29,-,6H5M,17,0;')]
r004  0        #0      16     30      6M14N5M      *      0        0        ATAGCTTCAGC      None      []
r003  2064     #0      29     17      6H5M      *      0        0        TAGGC      None      [('SA', 'ref,9,+,5S6M,30,1;')]
r001  147     #0      37     30      9M      #0      7        -39     CAGCGGCAT      None      [('NM', 1)]
```



SAM文件操作

选择指定区域

```
function AlignmentFile.fetch(self, contig=None, start=None, stop=None, ...)
```

Returns: An iterator over a collection of reads.

Return type: IteratorRow

```
import pysam

samfile = pysam.AlignmentFile("test.bam", "rb")

reads=samfile.fetch(contig='ref', start=0, end=15)
for read in reads:
    print(read)
```

```
>>> reads=samfile.fetch(contig='ref', start=0, end=15)
>>> for read in reads:
...     print(read)
...
r001  99      #0    7    30    8M2I4M1D3M    #0    37    39    TTAGATAAAGGATACTG    None    []
r002  0        #0    9    30    3S6M1P1I4M    *    0    0    AAAAGATAAGGATA    None    []
r003  0        #0    9    30    5S6M    *    0    0    GCCTAAGCTAA    None    [('SA', 'ref,29,-,6H5M,17,0;')]
```



SAM文件操作

文件输出

```
import pysam

samfile=pysam.AlignmentFile("test.bam", "rb")
paired_samfile=pysam.AlignmentFile("test_paired.bam", "wb", template=samfile)

for read in samfile.fetch():
    if read.is_paired:
        paired_samfile.write(read)

paired_samfile.close()
samfile.close()
```

```
(base) [zyzhao@sg59 test]$ samtools view test_paired.bam
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT * NM:i:1
```

```
(base) [zyzhao@sg59 test]$ samtools view -f 3 test.bam
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT * NM:i:1
```



SAM文件操作

堆积(pileup)格式

```
Coord      12345678901234  5678901234567890123456789012345
ref        AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1    TTAGATAAAGGATA*CTG
+r002      aaaAGATAA*GGATA
+r003      gcctaAGCTAA
+r004      ATAGCT.....TCAGC
-r003      ttagctTAGGC
-r001/2    CAGCGGCAT
```

```
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA *
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC *
r001 147 ref 37 30 9M = 7 -39 CAGCGGCAT *
```

```
ref 7 T 1 ^?.
ref 8 T 1 .
ref 9 A 3 .^?.^?.
ref 10 G 3 ...
ref 11 A 3 ..C
ref 12 T 3 ...
ref 13 A 3 ...
ref 14 A 3 .+2AG.+1G.$
ref 15 G 2 ..
ref 16 A 3 ..^?.
ref 17 T 3 ...
ref 18 A 3 .-1G.$
ref 19 G 2 *.
ref 20 C 2 ..
ref 21 T 2 ..
ref 22 G 2 .>
ref 23 T 1 >
ref 24 G 1 >
ref 25 C 1 >
ref 26 T 1 >
ref 27 A 1 >
ref 28 G 1 >
ref 29 T 2 >^2,
ref 30 A 2 >,
```



SAM文件操作

堆积(pileup)格式

```
function AlignmentFile.pileup(self,  
    contig=None, start=None, stop=None, ...)
```

Returns: an iterator over genomic positions.
Return type: IteratorColumn (PileupColumn)

```
for pileupcolumn in samfile.pileup():  
    print(pileupcolumn.reference_name,  
          pileupcolumn.reference_pos,  
          pileupcolumn.nsegments)
```

parameters: **pysam.PileupColumn.pileups**
list of reads (**pysam.PileupRead**) aligned to this column

```
class pysam.PileupRead()  
    Representation of a read aligned to a particular position in the reference sequence.
```

```
>>> for pileupcolumn in samfile.pileup():  
...     print(pileupcolumn.reference_name,  
...           pileupcolumn.reference_pos,  
...           pileupcolumn.nsegments)  
...  
ref 6 1  
ref 7 1  
ref 8 3  
ref 9 3  
ref 10 3  
ref 11 3  
ref 12 3  
ref 13 3  
ref 14 2  
ref 15 3  
ref 16 3  
ref 17 3
```



SAM文件操作

堆积(pileup)格式

```
import pysam

samfile=pysam.AlignmentFile("test.bam", "rb" )

for pileupcolumn in samfile.pileup("ref"):
    ref_name=pileupcolumn.reference_name
    ref_pos=pileupcolumn.reference_pos+1
    coverage=pileupcolumn.nsegments
    print("\ncoverage at %s:%s = %s" %
          (ref_name, ref_pos, coverage))

    for pileupread in pileupcolumn.pileups:
        if not pileupread.is_del and not pileupread.is_refskip:
            read=pileupread.alignment
            read_name=read.query_name
            base=read.query_sequence[pileupread.query_position]
            print('\tbase in %s = %s' % (read_name,base))
```

```
coverage at ref:7 = 1
    base in r001 = T

coverage at ref:8 = 1
    base in r001 = T

coverage at ref:9 = 3
    base in r001 = A
    base in r002 = A
    base in r003 = A

coverage at ref:10 = 3
    base in r001 = G
    base in r002 = G
    base in r003 = G

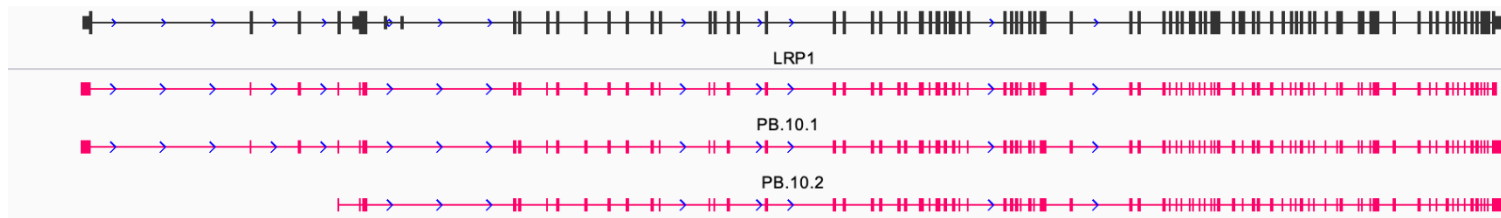
coverage at ref:11 = 3
    base in r001 = A
    base in r002 = A
    base in r003 = C

coverage at ref:12 = 3
    base in r001 = T
    base in r002 = T
    base in r003 = T
```



SAM文件操作

计算序列比对覆盖度



```
import pysam
import pandas as pd
cigar_type_dict = {0:'M',1:'I',2:'D',3:'N',4:'S',5:'H',6:'P',7:'=',8:'X',9:'B'}

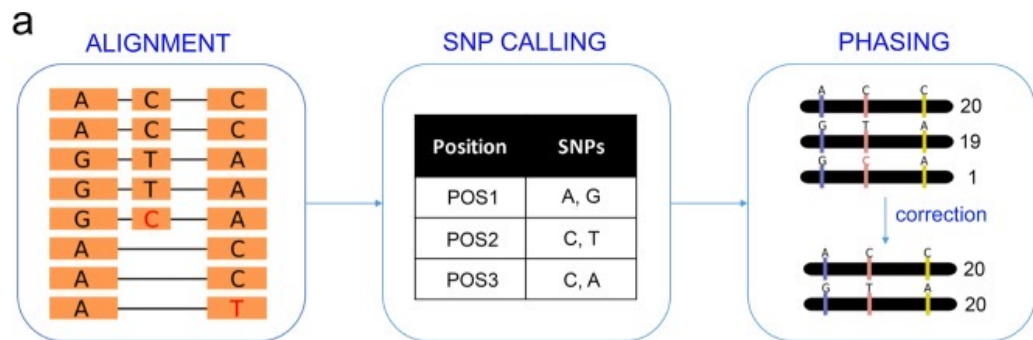
samfile = pysam.AlignmentFile('test.bam', 'rb')
qlens = pd.read_table('qlens.txt', names = ['qname', 'qlen']).set_index('qname')
for record in samfile:
    qry_name = record.query_name
    qry_len = qlens['qlen'][qry_name]

    if record.is_mapped:
        qry_start = 0
        qry_aln_len = 0
        cigar = pd.DataFrame(record.cigartuples, columns=['cigar_typeid', 'cigar_count'])
        cigar['cigar_type'] = cigar['cigar_typeid'].map(lambda x:cigar_type_dict[x])
        if cigar['cigar_type'][0] in ['S','H']:
            qry_start += cigar['cigar_count'][0]
        qry_aln = cigar[cigar['cigar_type'].isin(['M','I','=','X'])]
        if not qry_aln.empty:
            qry_aln_len += sum(qry_aln['cigar_count'])
        qry_end = qry_start + qry_aln_len
        qry_cover = (qry_end - qry_start) / qry_len
```



SAM文件操作

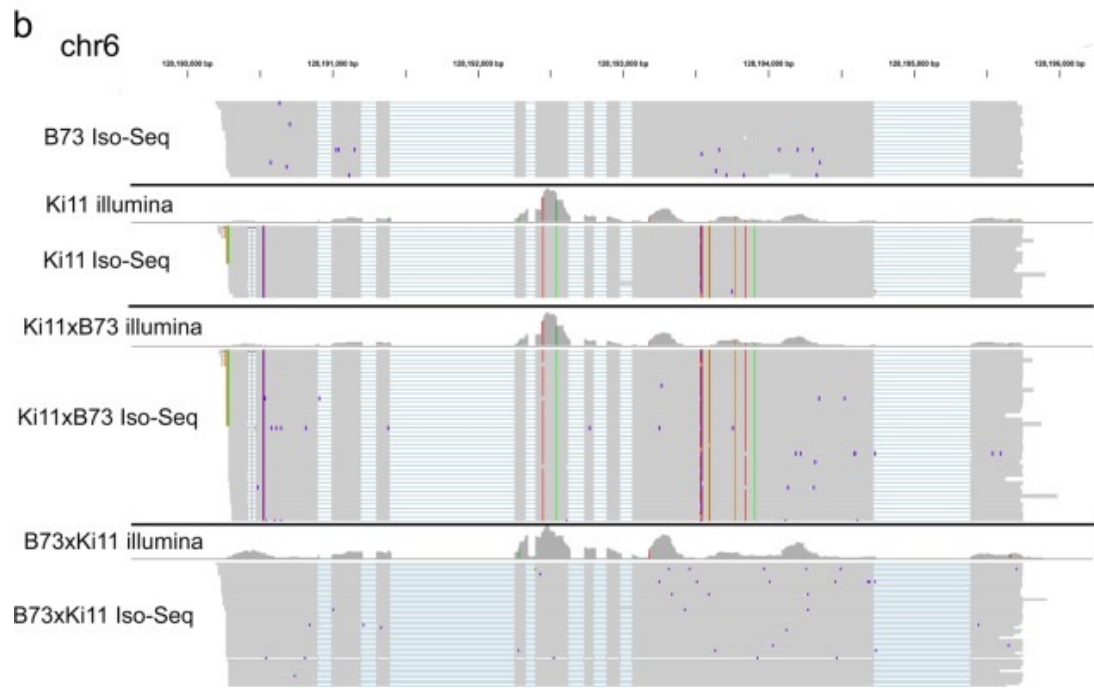
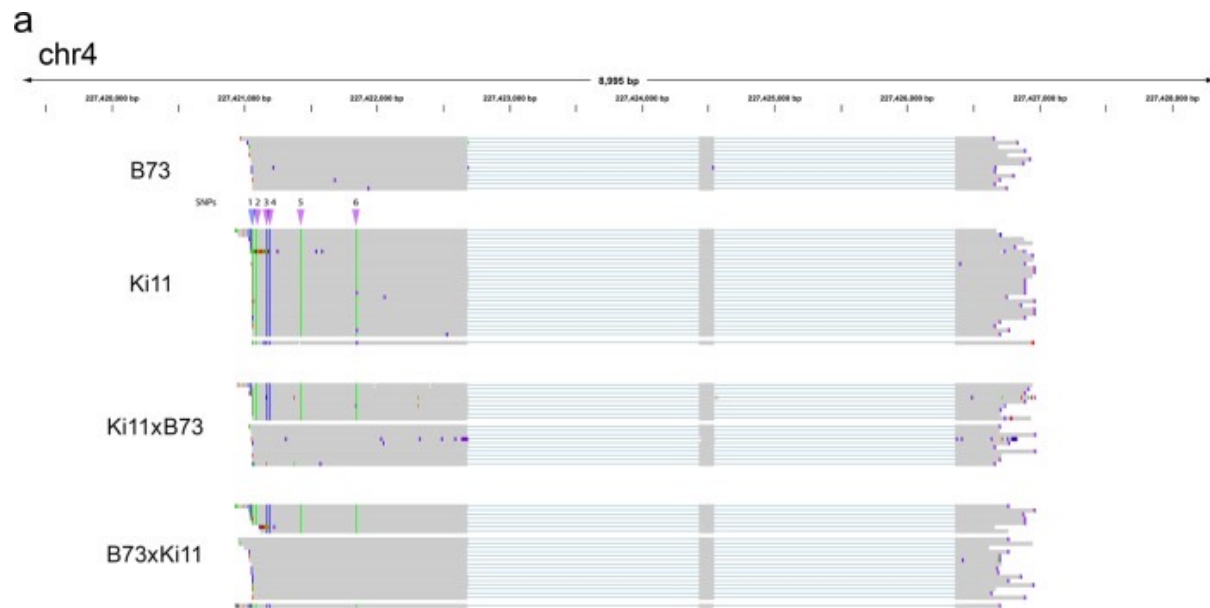
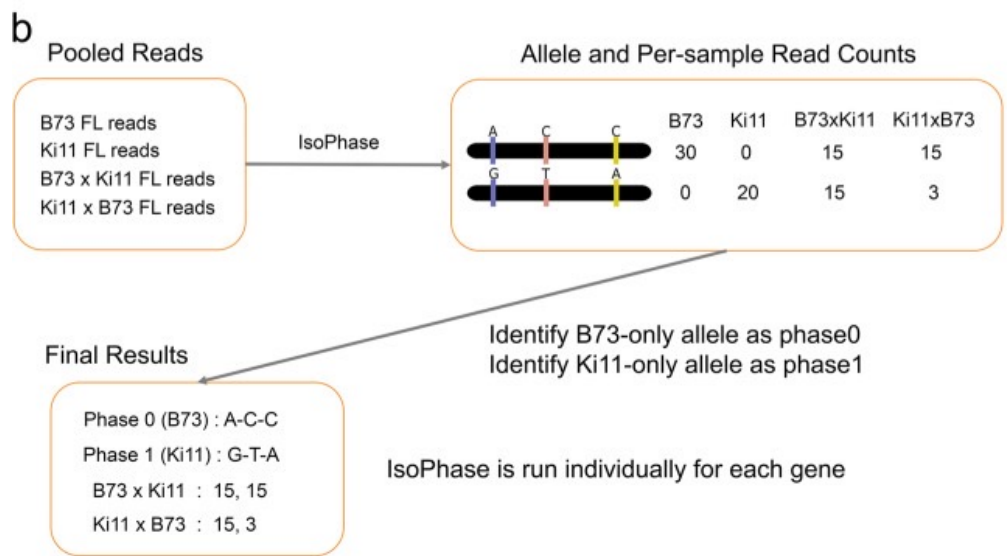
SNP鉴定及获取序列基因型



VCF OUTPUT

```

##fileformat=VCFv4.2
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT ISOFORM1 ISOFORM2
chr1 105 . A G . PASS DP=40;AF=0.50 GT:HQ 0|1:20,20 0:15
chr1 190 . C T . PASS DP=40;AF=0.50 GT:HQ 0|1:20,20 0:15
chr1 336 . C A . PASS DP=40;AF=0.50 GT:HQ 0|1:20,20 0:15
    
```



THANKS