二代测序数据：

比对：

1. bwa mem -t 10 -R "@RG\tID:E.smithi-1\tPL:illuminatLB:E.smithi-1\tSM:E.smithi-1" ref.fasta sample.1.fq.gz sample.2.fq.gz | samtools view -Sb - > ./01.bam/E.smithi-1.bam

2. samtools sort Hu25.bam -o Hu25.sorted.bam && samtools index Hu25.sorted.bam

一、Delly

1.安装：conda install -c bioconda delly

2.对每个个体call sv（for循环，echo）

delly call -g $reference -o $name.bcf $name.bam

1. 合并所有个体产生的sv文件

delly merge -o all.sites.bcf $name1.bcf $name2.bcf $name3.bcf ...

1. 对所有的位点每个个体call genotype（for循环，echo）

delly call -g $reference -v all.sites.bcf -o $name1.geno.bcf $name1.bam

1. 将每个个体的geno.bcf文件转成geno.vcf文件

for file in \*geno.bcf; do bcftools view $file -O v > $file.vcf; done

1. 使用脚本过滤掉IMPRECISE和LowQual的SV

for file in \*geno.bcf.vcf; do python3 fillter.py $file; done

#!/usr/bin/python3

import sys

import re

infile = sys.argv[1]

name = re.search(r'(.\*)\.vcf',infile)[1]

outfile = name + '.filter.vcf'

o = open(outfile, 'w')

with open(infile, 'r') as f:

for line in f:

line = line.strip()

if line.startswith('#'):

# print(line)

o.write(line + '\n')

continue

content = line.split()

if content[6] != "PASS" or re.match(r'IMPRECISE', content[7]):

continue

# print(line)

o.write(line + '\n')

1. lumpy（smoove）
2. 安装：需要python2.7的环境

conda install -c bioconda lumpy-sv

github安装smoove，添加环境变量，查看smoove需要的其他模块，全部安装完成。

1. 每个个体call sv（for循环，echo）

smoove call --outdir results-smoove/ --name $sample --fasta $reference -p 1 --genotype $name1.bam

1. 合并所有个体的vcf结果

smoove merge --name merged -fasta $reference ./results-smoove/\*.genotyped.vcf.gz

1. 对每个位点的所有个体call genotype（for循环，echo）

smoove genotype -d -x -p 1 --name $name1-joint --outdir results-genotyped/ --fasta $reference --vcf merged.sites.vcf.gz $name1.bam

1. 脚本过滤掉QUAL=0的SV

for file in \*genotyped.vcf; do python3 fillter.py $file; done

if content[5] == "0":

1. Manta
2. github安装Manta，添加环境变量后运行
3. 对全部个体call sv

for file in {11-AS2-1,13AS2-1,16-AS2-1}

do

echo "/data/01/user186/sv/software/manta-1.6.0.centos6\_x86\_64/bin/configManta.py --bam name1.bam --referenceFasta cishu.LG.fasta --runDir $file" >> run.sh

done

每个样本会生成一个文件夹，文件夹下有一个runWorkflow.py，批量并行每个文件夹中的脚本

for file in {11-AS2-1,13AS2-1,16-AS2-1}

do

echo "python /data/01/user191/svcishu/manta/$file/runWorkflow.py" >> run1.sh

done

结果：每个文件夹下results/variants/diploidSV.vcf

1. 对每个结果文件过滤，过滤掉QUAL小于20的SV

先解压后过滤（for循环解压过滤）

if content[5] < "20":

1. 为了后期合并方便，把结果文件批量改名

for file in {11-AS2-1,13AS2-1,16-AS2-1,AS1-21}

do

mv /data/01/user191/svcishu/manta/$file/results/variants/diploidSV.filter.vcf /data/01/user191/svcishu/manta/$file/results/variants/$file.filter.vcf

Done

1. 三个软件结果的合并和处理
2. svimmer合并

for i in \*.vcf;do bgzip $i;done

for i in \*.vcf.gz;do tabix -p vcf $i;done

svimmer --threads 10 all.vcf.list LG01 LG02 LG03 LG04 ......> input.vcf (生成的input.gz也需要用bgzip压缩，并且用tabix建立索引)

all.vcf.list中包括三个软件所有的过滤后的vcf结果文件

find .-name “\*.filter.vcf”

染色体名称必须在参考基因组的vcf文件中提取（awk）

1. graphtyper再次给SV分类型

./graphtyper genotype\_sv reference.fasta input.vcf.gz --sams=bam.list --region=LG01:start-end

写脚本分别生成每条命令行：

import os

import sys

f = open("chrpos.bed","r")

for line in f:

line = line.strip()

print("graphtyper genotype\_sv cishu.LG.fasta merge.vcf.gz --sams=bam.list --region=" + line)

f.close()

脚本中使用的chrpos.bed需要自己生成

awk '{print $1, 1, $2}' cishu.LG.fasta.fai > chrpos.bed

bam.list中包括所有需要合并的个体的bam文件

1. 合并每条染色体的vcf文件

方法一：gatk MergeVcfs -I concat-a.vcf -I concat-b.vcf -O combine\_gatk.vcf

vcf.list中包含所有文件夹中的结果文件vcf.gz

cat vcf.list | awk '{print $1,$2="-I"}' | awk '{print $2,$1}' | tr "\n" " "

方法二：grep '^#' chr1.vcf > merge.vcf

grep -v '^#' chr1.vcf chr2.vcf chr3.vcf chr4.vcf >> merge.vcf

1. 查看合并后的combine\_gatk.vcf文件（merge），发现有的SV有一条AGGREGATED,BREAKPOINT，使用bcftools删除掉BREAKPOINT这条

bcftools view --include "SVMODEL='AGGREGATED'" -Oz -o get\_agg.vcf.gz combine\_gatk.vcf

1. 使用vcftools过滤掉GT的max missing0.95的SV

vcftools --vcf get\_agg.vcf --max-missing 0.95 --recode --recode-INFO-all --out vcftools-filliter

1. 过滤假阳性区域，提取长度和类型

写脚本过滤掉SVSIZE大于2000的假阳性SV（vcf），同时提取type和length（txt）

/data/01/user186/svcishu/python/04.getlength.py

/data/01/user186/svcishu/python/kzr.filter.lengtn.py

写脚本提取每个样本的SV类型

/data/01/user186/svcishu/python/05.single.sample.py

1. 计算SV的FST值，不分窗口，按照位点计算

vcftools --vcf test.vcf --weir-fst-pop 1\_population.txt --weir-fst-pop 2\_population.txt --out p\_1\_2—single

1. 功能富集（二代SV）
2. 取5%高分化区域

POS前后加减2000

cat 0.05Fst.txt | awk '{print $1,$2}'| awk '{print $1,$2-2000,$2+2000}' | tr " " "\t" > 0.05fst.bed

1. 与注释文件区域取重叠

bedtools intersect -wo -F 0.1 -a 0.05fst.bed -b cishu.genome.gff | awk -F "\t" '{print $4"\t"$7"\t"$8"\t"$10"\t"$12}' | sort -u > 0.05.top.gene

1. 写脚本提取.top.gene中的evm编号

/data/01/user186/svcishu/python/evm.py

python3 evm.py 0.05.top.gene | awk 'BEGIN{FS=".";OFS="."}{print $1,$2,$3,$4}' > 0.05evm.list

1. 从gff3文件中提取编号和GO富集编号

/data/01/user186/svcishu/python/from\_getevm.py

1. 将两个文件放到TBTOOLS中功能富集，画图

**Vcf文件处理合并：**

python **01.getlength.py** vcftools-filter.recode.vcf

最后生成Sheep.filter.vcf

~~根据chr和pos排序（~~[~~https://www.zxzyl.com/archives/741~~](https://www.zxzyl.com/archives/741)~~）~~

~~java -jar /data/00/software/picard/picard.2.21.5.jar SortVcf I=Sheep.filter.vcf O=Sheep.filter.sorted.vcf~~

**~~Sheep.filter.sorted.vcf~~**

不排序！

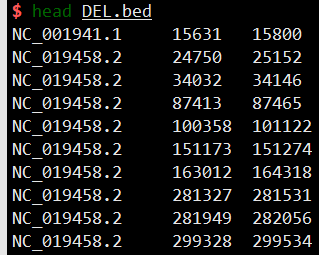
**bedtools提取vcf文件中DEL 和INS的fastq**

其中INS的fastq序列已经存在在vcf中

提取DEL的序列：

bedtools getfasta -fi Texel.fna -bed DEL.bed > DEL.fastq

bed文件： chr, start, end



Texel.fna: 参考基因组

用RepeatMasker软件注释DEL和INS：

$ cat DEL.sh

/data/00/software/singularity/singularity\_built/bin/singularity exec -B /data -B /home /data/00/software/singularity/TETools RepeatMasker -pa 5 -species human -nolow -no\_is -norna -q -dir ./DEL\_repeatmasker\_repeat **DEL.fastq** 1>reptmasker.log.o.txt 2>reptmasker.log.e.txt

$ cat INS.sh

/data/00/software/singularity/singularity\_built/bin/singularity exec -B /data -B /home /data/00/software/singularity/TETools RepeatMasker -pa 5 -species human -nolow -no\_is -norna -q -dir ./INS\_repeatmasker\_repeat INS.fastq 1>reptmasker.log.o.txt 2>reptmasker.log.e.txt

**SV基因注释**

perl -alne 'if ($F[2] eq "gene") {$F[8] =~ /ID=(.\*?);/; $name = $1; $start = $F[4] - 1; $end = $start + **20001**; print "$F[0]\t$start\t$end\t$name";}' GCF\_000298735.2\_Oar\_v4.0\_genomic.gff | sort -k1,1 -k2,2n -k3,3n | bgzip -c > Texel.genes.downstream.bed.gz

perl -alne 'if ($F[2] eq "gene") {$F[8] =~ /ID=(.\*?);/; $name = $1; $start = $F[3] - **20001**; $end = $F[3]; print "$F[0]\t$start\t$end\t$name";}' GCF\_000298735.2\_Oar\_v4.0\_genomic.gff | sort -k1,1 -k2,2n -k3,3n | bgzip -c > Texel.genes.upstream.bed.gz

perl -alne 'if ($F[2] eq "exon") {$F[8] =~ /ID=(.\*?);/; $name = $1; $start = $F[3] - 1; $end = $F[4] - 1; print "$F[0]\t$start\t$end\t$name";}' GCF\_000298735.2\_Oar\_v4.0\_genomic.gff | sort -k1,1 -k2,2n -k3,3n | bgzip -c > Texel.exons.srt.bed.gz

perl -alne 'if ($F[2] eq "gene") {$F[8] =~ /ID=(.\*?);/; $name = $1; $start = $F[3] - 1; $end = $F[4] - 1; print "$F[0]\t$start\t$end\t$name";}' GCF\_000298735.2\_Oar\_v4.0\_genomic.gff | sort -k1,1 -k2,2n -k3,3n | bgzip -c > Texel.genes.srt.bed.gz

perl -alne 'if ($F[2] eq "CDS") {$F[8] =~ /ID=(.\*?);/; $name = $1; $start = $F[3]; $end = $F[4]; print "$F[0]\t$start\t$end\t$name";}' GCA\_017524585.1\_CAU\_O.aries\_1.0\_genomic.gff | sort -k1,1 -k2,2n -k3,3n | bgzip -c > Tibetan.cds.srt.bed.gz

python cds.py | bgzip -c > Tibetan.cds\_end.srt.bed.gz

python cds.py | bgzip -c > Tibetan.cds\_start.srt.bed.gz

2. 建立索引

tabix Tibetan.genes.upstream.bed.gz

tabix Tibetan.genes.downstream.bed.gz

tabix Tibetan.cds\_end.srt.bed.gz

tabix Tibetan.cds\_start.srt.bed.gz

tabix Tibetan.exons.srt.bed.gz

3. 注释

# sv number

./vcfanno\_linux64 config.toml ../Sheep.filter.vcf > **Sheep\_anno.vcf**

Fst 计算

vcftools --vcf ../gene\_anno/Sheep\_anno.vcf --weir-fst-pop low\_sheep --weir-fst-pop high\_Tibetan --out Sheep

Rscript fstTop0.05.R

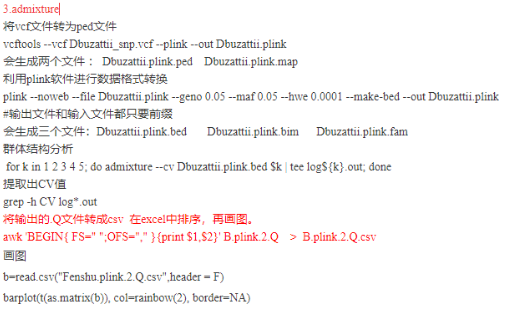
python 00.GetFstGene.py ../gene\_anno/sv\_genes fsttop0.05.txt > **GetFstGene.txt**

PCA:

vcftools --vcf ../Sheep\_anno.vcf –plink --out Sheep\_anno

plink –file Sheep\_anno --pca

Structure:



STRUCTURE:

plink --make-bed --geno 0.05 --maf 0.05 --hwe 0.01 --file Sheep\_anno --out ./structure/Sheep\_anno

for k in 1 2 3 4 5;do admixture --cv Sheep\_anno.bed $k| tee log${k}.out;done

grep -h "CV" log\*.out (选择CV最小的，可分的最好！)

CV error (K=1): 0.59358

CV error (K=2): 0.67060

CV error (K=3): 0.76209

CV error (K=4): 0.87673

CV error (K=5): 1.03203

awk 'BEGIN{FS=" ";OFS=","}{print $1,$2,$3,$4,$5}' Sheep\_anno.2.Q > Sheep\_anno.2.Q.csv

cat Sheep\_anno.2.Q.csv| sort -t "," -k 1n -k 2n -k 3n -k 4n > Sheep\_anno.2.Q.sorted.csv

Rscript structure.R

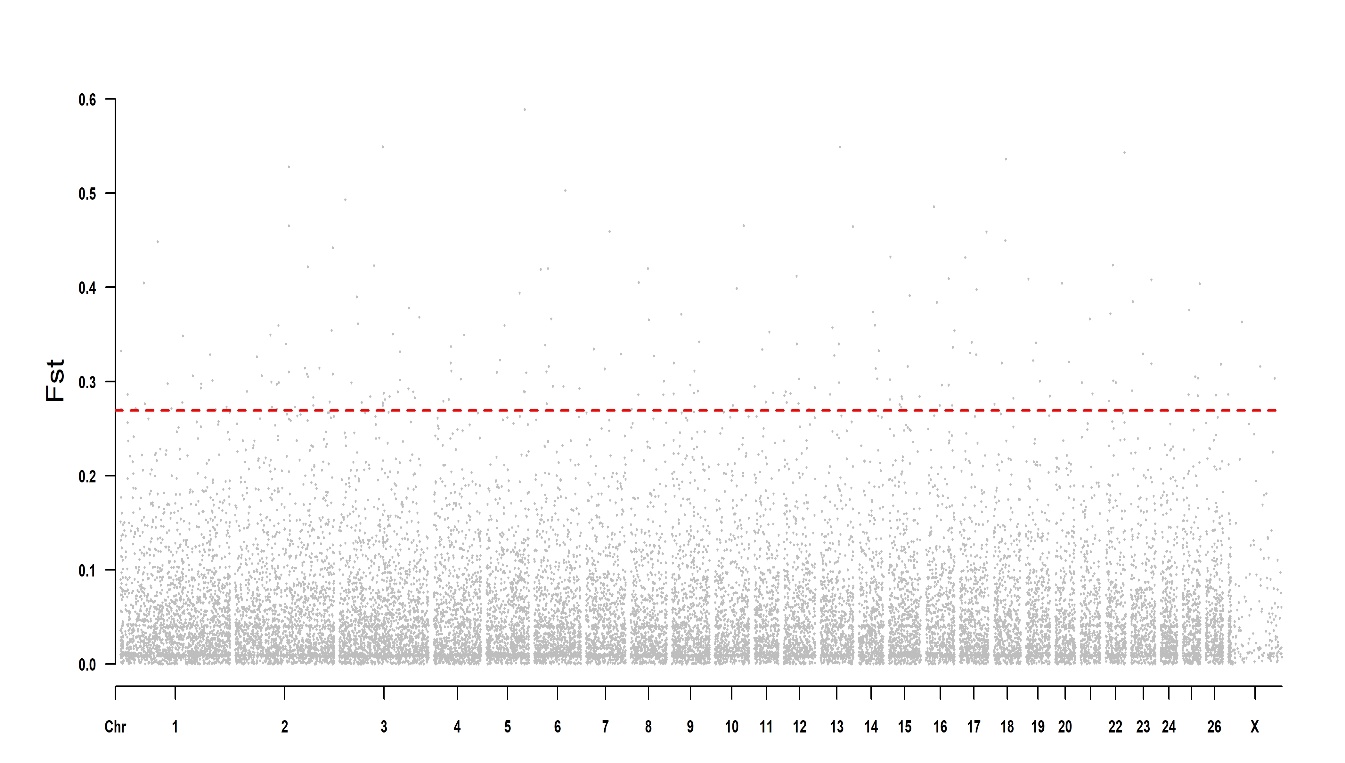
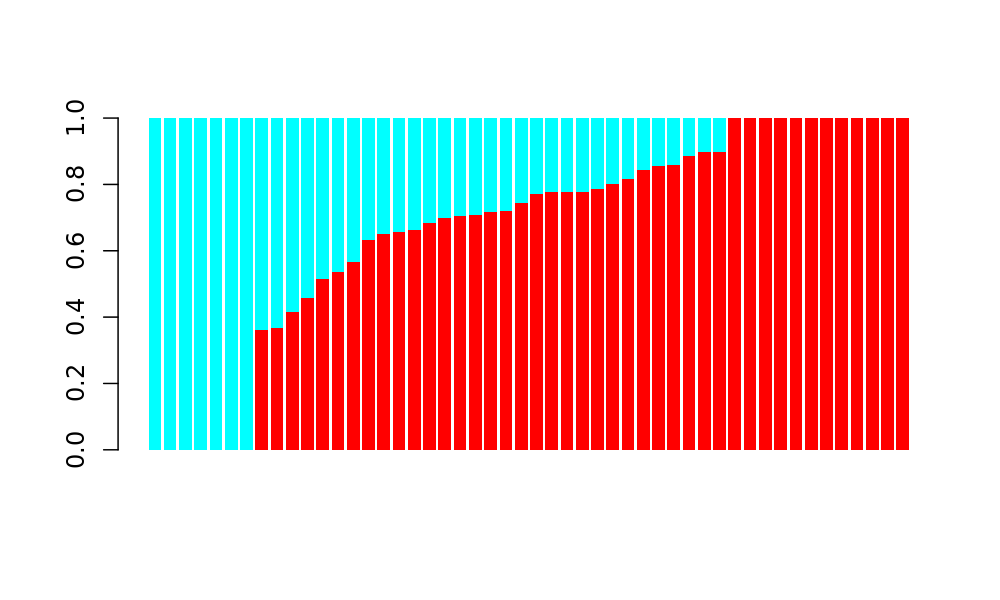
画图











K=2