实验记录 hisat2比对

使用hisat2构建索引：

hisat2本身的优势是其更适合用于转录组RNA Seq数据，因为RNA由于可变剪接，其逆转录生成的cDNA会有本身属于参考序列的一部分，但是因为其中间缺少内含子而比对失败的事情，而hisat2设置了对于这些序列进行切开操作并重新比对，寻找其内含子位置。

对于hisat2软件，构建索引需要额外的注释文件，没有的话也可以构建：

```
hisat2-build -p 10 genome.fa genome
```

其中genome是索引的名字，genome.fa是参考序列的名字,最终会生成八个.ht2文件

使用hisat2进行比对，转换成bam格式，并且进行排序和构建索引：

```
hisat2 -p 10 -x ~/reference/my.ref.fa -1 SRR6023853_1_filtered_1.fastq -2
SRR6023853_1_filtered_2.fastq -S SRR6023853.sam && samtools view -bS
SRR6023853.sam > SRR6023853.bam && samtools index SRR6023853.bam
```

最终会生成对应的.bam文件和.bam.bai文件

注意一点的是，这一步生成的bam文件是没有Read group的，这将导致gatk软件无法处理，后面需要用picard处理一下

```
AddOrReplaceReadGroups --INPUT
/user/WH/workspace/train_data_test/2.callsnp_gatk/1.markduplicates/Mark_result_pa
rt/SRR6023853_Markdup.bam --OUTPUT ./SRR6023853_ADD.bam --RGLB tLB_SRR6023853 --
RGPL ILLUMINA --RGPU unit1 --RGSM tSM_SRR6023853 --RGID 1 --VERBOSITY INFO --
QUIET false --VALIDATION_STRINGENCY STRICT --COMPRESSION_LEVEL 5 --
MAX_RECORDS_IN_RAM 500000 --CREATE_INDEX false --CREATE_MD5_FILE false --
GA4GH_CLIENT_SECRETS client_secrets.json --help false --version false --
showHidden false --USE_JDK_DEFLATER false --USE_JDK_INFLATER false
```

```
Required Arguments:

--INPUT,-I <String>           Input file (BAM or SAM or a GA4GH url).  Required.

--OUTPUT,-O <File>            Output file (SAM, BAM or CRAM).  Required.

--RGLB,-LB <String>           Read-Group library  Required.

--RGPL,-PL <String>           Read-Group platform (e.g. ILLUMINA, SOLID)
Required.

--RGPU,-PU <String>           Read-Group platform unit (eg. run barcode)
Required.

--RGSM,-SM <String>           Read-Group sample name  Required.
```

```
picard AddOrReplaceReadGroups -I
/user/WH/workspace/train_data_test/2.callsnp_gatk/1.markduplicates/Mark_result_pa
rt/SRR6023853_Markdup.bam -O ./SRR6023853_ADD.bam --RGLB tLB_SRR6023853 --RGPL
ILLUMINA --RGPU unit1 --RGSM tSM_SRR6023853
```

实验记录 Markduplicates

qPCR测序会由于扩增差异导致DNA的重复序列数量不同，这样会影响我们后续对SNP数量的统计，因此要进行标记PCR重复操作。

操作的原理是gatk理由SAM文件中的信息确定比对序列的来源情况，进而判断其是否为重复序列(Mark duplicates)

```
gatk MarkDuplicates --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP 1000 --INPUT
/user/WH/workspace/train_data_test/1.callSNP_samtools/2.mapping/sam2bam/1.sort+in
dex/sort_bam/SRR6023885_1_filtered_1.sortP.bam --OUTPUT ./SRR6023885_Markdup.bam
--METRICS_FILE ./metrics.txt --MAX_SEQUENCES_FOR_DISK_READ_ENDS_MAP 50000 --
SORTING_COLLECTION_SIZE_RATIO 0.25 --TAG_DUPLICATE_SET_MEMBERS false --
REMOVE_SEQUENCING_DUPLICATES false --TAGGING_POLICY DontTag --CLEAR_DT true --
DUPLEX_UMI false --FLOW_MODE false --FLOW_QUALITY_SUM_STRATEGY false --
USE_END_IN_UNPAIRED_READS false --USE_UNPAIRED_CLIPPED_END false --
UNPAIRED_END_UNCERTAINTY 0 --FLOW_SKIP_FIRST_N_FLOWS 0 --FLOW_Q_IS_KNOWN_END
false --FLOW_EFFECTIVE_QUALITY_THRESHOLD 15 --ADD_PG_TAG_TO_READS true --
REMOVE_DUPLICATES false --ASSUME_SORTED false --DUPLICATE_SCORING_STRATEGY
SUM_OF_BASE_QUALITIES --PROGRAM_RECORD_ID MarkDuplicates --PROGRAM_GROUP_NAME
MarkDuplicates --READ_NAME_REGEX <optimized capture of last three ':' separated
fields as numeric values> --OPTICAL_DUPLICATE_PIXEL_DISTANCE 100 --
MAX_OPTICAL_DUPLICATE_SET_SIZE 300000 --VERBOSITY INFO --QUIET false --
VALIDATION_STRINGENCY STRICT --COMPRESSION_LEVEL 2 --MAX_RECORDS_IN_RAM 500000 --
CREATE_INDEX false --CREATE_MD5_FILE false --help false --version false --
showHidden false --USE_JDK_DEFLATER false --USE_JDK_INFLATER false
```

做这一步的时候我用的最新版的gatk4，其中--REMOVE_DUPLICATES false设置的有点问题，导致我的运算时间很长，所以建议参数设置如下(使用gatk)：

```
nohup $java $gatk MarkDuplicates -I
/user/WH/workspace/train_data_test/1.callSNP_samtools/2.mapping/sam2bam/1.sort+in
dex/sort_bam/SRR6023883_1_filtered_1.sortP.bam -O ./SRR6023883_Markdup.bam -M
metrics.txt   --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP 1000 > gatk_mkdup.log 2>&1 & #
这一步时间会很长
```

运行会生成 SRR6023883_Markdup.bam 和 metrics.txt

-M metrics.txt设置输出报告，其中包含有关重复标记过程的统计信息。

标记完成之后就可以构建索引文件了：

使用GATK callSNPs需要的索引文件有三个：**reference的.fai、reference的.dict、merged_bam_markdup.bam文件的.bai文件。**

```
samtools index SRR6023853_Markdup.bam

samtools faidx $ref

samtools CreateSequenceDictionary -R $ref -O my.ref.dict
```

实验记录 call indel

使用gatk进行indel calling：

```
gatk3 -nt 25 -R /user/WH/reference/my.ref.fa -T RealignerTargetCreator -o
./SRR6023853.realn.intervals -I
/user/WH/workspace/train_data_test/2.callsnp_gatk/1.markduplicates/Mark_result_pa
rt/SRR6023853_ADD.bam
```

报错：

```
##### ERROR MESSAGE: Unsupported CIGAR operator N in read SRR6023853.16828323 at
comp1892_c0_seq1:48. If you are working with RNA-Seq data, see
https://software.broadinstitute.org/gatk/documentation/article?id=3891 for
guidance. If you choose to disregard those instructions, or for other uses, you
have the option of either filtering out all reads with operator N in their CIGAR
string (add --filter_reads_with_N_cigar to your command line) or overriding this
check (add -U ALLOW_N_CIGAR_READS to your command line). Notice however that the
latter is unsupported, so if you use it and encounter any problems, the GATK
support team not be able to help you.
```

是因为gatk不支持转录组数据，所以需要运行以下命令：

```
gatk3 -nt 25 -R /user/WH/reference/my.ref.fa -T RealignerTargetCreator -o
./SRR6023853.realn.intervals -I ./SRR6023853_ADD.bam --filter_reads_with_N_cigar
```

这里面SRR6023853_ADD.bam文件是SRR6023853_Markdup.bam文件使用picard添加read group之后的文件。在使用bwa的时候我们可以直接在比对的时候进行添加，而此次我们使用的是hisat2进行比对，这个软件没有这个功能，因而这个步骤是必须的，从而生成了.intervals文件，其中记录了比对结果的indel信息。

之后的call indel也需要这样做，加上--filter_reads_with_N_cigar：

```
gatk3 -nt 25 -R /user/WH/reference/my.ref.fa -T IndelRealigner --targetIntervals
./SRR6023853.realn.intervals -o ./SRR6023853.realn.bam -I ./SRR6023853_ADD.bam --
filter_reads_with_N_cigar
```

这里报错显示使用IndelRealigner的时候不能设置-nt，所以把他去掉

```
gatk3 -R /user/WH/reference/my.ref.fa -T IndelRealigner --targetIntervals
./SRR6023853.realn.intervals -o ./SRR6023853.realn.bam -I ./SRR6023853_ADD.bam --
filter_reads_with_N_cigar
```

这样我们就完成了对indel区域进行重新比对，这一过程耗时较久。

实验记录 call SNP

```
samtools mpileup -f /user/WH/reference/my.ref.fa ./SRR602853.realn.bam >
SRR6023853.txt
```

```
samtools mpileup -g -t DP,DP4,SP -f /user/WH/reference/my.ref.fa -o
./SRR6023853.bcf ./SRR602853.realn.bam
```

```
bfctools call -cvM -o SRR602853_raw.vcf ./SRR602853.bcf
```

```
vcfutil.pl varFilter SRR602853_raw.vcf > SRR602853_filtered.vcf
```

最终我们的SRR602853_filtered.vcf文件就是call 出来的variants

vcf文件解读：

```
comp1036396_c0_seq1#CHORM
237#POS
.#ID
G#REF
A#ALT
129.514#QUAL
.#FILTER
DP=5;VDB=0.459448;SGB=-0.556411;MQSB=1;MQ0F=0;AF1=1;AC1=2;DP4=0,0,1,3;MQ=60;FQ=-3
8.9864#INFO
GT:PL:DP:SP:DP4#FORMAT
1/1:162,12,0:4:0:0,0,1,3#SAMPLEs
```

```
comp436591_c0_seq1#CHORM
175#POS
.#ID
CCTTCTTCT#REF
CCTTCT#ALT
53.4663#QUAL
.#FILTER
INDEL;IDV=2;IMF=0.5;DP=4;VDB=0.348933;SGB=-0.453602;MQSB=1;MQ0F=0;AF1=0.5;AC1=1;D
P4=2,0,1,1;MQ=60;FQ=55.6861;PV4=1,0.074601,1,1
#INFO
GT:PL:DP:SP:DP4 #FORMAT
0/1:91,0,98:4:0:2,0,1,1#SAMPLEs
```

这是我们得到的raw.vcf的文件中的一个SNP和一个indel的信息，其中在SNP中，FILTER列表示过滤信息，PASS表示这个点可以称为一个SNP，而此处我们是raw.vcf，是还没有进行过滤标记的，所以这里是个.

SNP和indel的过滤：这里就使用gatk4了，下面的命令是对indel或者SNP的提取和过滤命令：

```
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
SelectVariants \
```

```
-R $REFERENCE \
-V all.merge.vcf \
--select-type SNP \
-O all.raw.snp.vcf 1>${logDir}/gatk.log 2>&1

### 过滤 SNP （ Filter 列加标记）
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
VariantFiltration \
-R $REFERENCE \
-V all.raw.snp.vcf \
--filter-expression "QD < 2.0 || MQ < 40.0 || FS > 60.0 \
|| SOR > 3.0 || MQRankSum < -12.5 || ReadPosRankSum < -8.0" \
--filter-name 'SNP_filter' \
-O all.filter.snp.vcf

### 提取过滤好的 SNP
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
SelectVariants \
-R $REFERENCE \
-V all.filter.snp.vcf \
--exclude-filtered \
-O all.filtered.snp.vcf

### 提取 INDEL
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
SelectVariants \
-R $REFERENCE \
-V all.merge.vcf \
--select-type INDEL \
-O all.raw.indel.vcf

### 过滤 INDEL （ Filter 列加标记）
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
VariantFiltration \
-R $REFERENCE \
-V all.raw.indel.vcf \
--filter-expression "QD < 2.0 || FS > 200.0 || SOR > 10.0 \
|| MQRankSum < -12.5 || ReadPosRankSum < -8.0" \
--filter-name 'INDEL_filter' \
-O all.filter.indel.vcf

### 提取过滤好的 INDEL
gatk \
--java-options "-Xmx10g -Djava.io.tmpdir=./tmp" \
SelectVariants \
-R $REFERENCE \
-V all.filter.indel.vcf \
--exclude-filtered \
-O all.filtered.indel.vcf
```

使用gatk4的时候会出现这个问题：

```
nohup gatk SelectVariants -R ~/reference/my.ref.fa -V SRR6023853_raw.vcf  --
select-type SNP -O SRR6023853_SNP_raw.vcf> SelectVariants.log  2>&1 &

A JNI error has occurred, please check your installation and try again
```

这是因为之前把.bashrc的jiava环境变量注释掉了，所以现在把他加上即可：

```
export JAVA_HOME=/user/WH/software/jdk-21.0.1
export PATH=$JAVA_HOME/bin:$PAT
```

之后会生成一个只包含SNP而去除了indel的vcf，以及一个对应的idx文件

注意这样我们得到的是raw的vcf，其中没有SNP的PASS信息，因而我们要先进行过滤操作。

SNP的过滤操作：

首先是对INFO进行过滤:

```
nohup gatk VariantFiltration -R ~/reference/my.ref.fa -V ./SRR6023853_SNP_raw.vcf
--filter-expression "QD < 2.0 || MQ < 40.0 || FS > 60.0 || SOR > 3.0 || MQRankSum
< -12.5 || ReadPosRankSum < -8.0" --filter-name 'SNP_filter' -O
SRR6023853.filter.snp.vcf > VariantFiltration.log 2>&1 &
```

提取所有的PASS：

```
nohup gatk SelectVariants -R ~/reference/my.ref.fa -V ./SRR6023853.filter.snp.vcf
--exclude-filtered -O SRR6023853.filtered.snp.vcf > selectSNP.log 2>&1 &
```

可以进一步去除10bp之内同时存在三个SNP的数据，以及indel 5bp内的SNP数据

```
#标出10bp范围3个SNP的 "SnpCluster“
nohup gatk VariantFiltration -V ./SRR6023853.filtered.snp.vcf -cluster 3 -window
10 -O ./SRR6023853.filter_10bp.snp.vcf > mark_10bp.log 2>&1 &
#去除上一步标出的SnpCluster"
nohup gatk SelectVariants -V SRR6023853.filter_10bp.snp.vcf -O
SRR6023853.filtered_10bp.snp.vcf -select "FILTER == SnpCluster" --invertSelect >
del_10bp.log 2>&1 &

nohup bcftools filter -g 5 -O v -o SRR6023853.filter_5bp.snp.vcf
./SRR6023853_raw.vcf > del_5bp_indel.log 2>&1 & #去除indel附近5bp范围内的SNP,这一步的
vcf要包含indel信息，所以这一步一定要先做
```

这里我选择去除indel 5bp内的SNP数据，最终完成过滤。

SNP过滤标准

建树的模型，参数

实验记录 buildtree

文件合并：

```
zcat ./*.vcf.gz
```

首先对所有的vcf进行合并操作，然后得到vcf_merge.pl.vcf.gz：

文件转换，vcf2phy

```
nohup python vcf2phylip.py -
-/user/WH/workspace/train_data_test/2.callsnp_gatk/4.filter_opteration/5.final_fi
lter_SNP_and_cat/vcf_merge.pl.vcf > vcf2phy.log 2>&1 &
```

输出文件为vcf_merge.pl.min1.phy

```
nohup iqtree -s ./vcf_merge.pl.min1.phy -keep-ident  -m GTR+F+R6+ASC -fast -nt 25
> build1.log 2>&1 &
```

文件转换，vcf2fa

```
perl vcf2fa.pl
/user/WH/workspace/train_data_test/2.callsnp_gatk/4.filter_opteration/5.final_fil
ter_SNP_and_cat/vcf_merge.pl.vcf.gz 57 > vcf.merge.fa
```

这两种转换的脚本都出现了一个问题，那就是只输出了一个样本的文件：

```
>tSM_SRR6023853
AATCYCYYGGAWCAGMYRSYTSMYTRGWMKTTRTATAGYRAGTMAGATTKTRRTCWTGGCYAYYGTATCCCGGRTGATGTY
TYAWGRRKTACTCGGTAAGACGACAGARKACGAAGCTCCCKTASCCGTGTAGTTGTCYRCAMCGMTGGTCGAGAACAWTGA
GTARCTGGTTTACCCCTGMATTCATMWAYMAKTGRWTAGTAGCTTTMAARTGCRGTAATCGTYRMYWMRYMAGTTAGTGAC
KYYGTCCRGCAGCRTTCCAGTCAACRRCRGSAACCCRMTTCAMGWCTAMRGKKMATKTRWKSTTACCTATYCYACKAYRWW
YCMGKSRMCAGTCTTCMCKTRKAARCGTTAAYGTYWKRYCAKCTMYYWRACAGCSGTCCGGRGMMAACTTWAGARCGSTGG
TCRTCYRCATCTTGSMRCYRKTYGATTTAACTTTGTTGTGSGGTCRGTYATCCTTGCYRMTATRKTYCTWGTMAATTCTTA
TAAYTGACAGMGATWYRKKAACYTTKMATYKTCGARSAATGCCMTCYTTTYYACACCGSGTGRCGTRKCTTCTTTRMYRYC
AMGWRYTWYRRAAYSTTYGCATCGCCRYYKTTTATGCWTCCTGAGKKAYCRMKYGYGCRAYAYTACSCYTRRCRRACTATG
MCTMKCGYTAGGGAAATCAAACAAGTAWGATRTYCACCTTRRATTYYCYMTGCGAAAAYWGAMRAMGKRRRTTTCYTGSCG
ACAGAAGTCYSRGYRTKGGGRYYRGTGACCCTGGCCKSASATCTAGGATGTTTTACGTGGATGTTGYYYYKMGGGGGGCTG
GGCAGACAAGCTRATAGCWYTMTWGCYRRAGYYRTYYTGRRGCTYTCCGRTMAGYRGATAWTGATGAAGRCTCGAKCTATC
AGCCACGAGCTATATATACYCRMCTYMYGAATGKSMYYCRCTTAGCCGCAAAWCCGCAAWRTYCGRCGATCGSRYTATRRR
YTRRCMTTAATAAKRCCCWTRTRTYYTMWCTTATGTTAYTTTCTTATAARTCTKCCACTYRYCAKAGTGGRMGATGGGTRT
YYRSYYGWTACCGGSCGYYTGGGCTTKCGYCTGSRRYTYRGGGGATTGCATMAMTRCYGTGGCGRCGCYCTGYCYSGAGCT
YMATYRGMAGKCYRTYGGCAWYAKRYYRMYATWYGGKYCYAGAGCGGTAATGAATACAATRAGGTCAAGTAGASCGAAGAA
ACAATATRCGAACTATTCTTCATTCCKMGWRCGCGWARYCGAWRTARCCCYGAGGCTATACACGACCTCAGTGAAGTTCTG
GTTGAGTCCAYRRRCAMAAG,ATGCAAARAYCCAGMGCCRCATCCGKGRCAAGAYCATGWCKGYCAGCWCYGACGTACGCW
AKRAYRARYWCYTRGYRCRTCSTKGMAAYRYTGGGYYGCMAGSRTYGARGKCYGTYRYMYYKMRTTTCTCGTCCCCGCGGA
CGAGTYSTATATTAGTCCCCTTCGTAYCATGCACA
```

```
1 3824645
tSM_SRR6023853   ATCCCYSTMRRGAGTGCGGTATCCCGGRTGATKYARYCTCGGTGGACAYKCCCRCCG
```

这样的文件是无法做出进化树的，因为只有一个sample的信息。检查合并得到的vcf文件，发现合并完的文件的头文件只包含第一个文件的信息，检查我们每个样本的vcf文件，结果是可以对应到的，所以我觉得是合并的时候出了问题。

更换另外一种方式进行合并，首先要知道合并是意味着什么？这里使用bcftools合并

```
bcftools merge ./SRR6023853.final.vcf.gz ./SRR6023854.final.vcf.gz
./SRR6023855.final.vcf.gz ./SRR6023856.final.vcf.gz ./SRR6023857.final.vcf.gz
./SRR6023858.final.vcf.gz ./SRR6023859.final.vcf.gz ./SRR6023860.final.vcf.gz
./SRR6023861.final.vcf.gz ./SRR6023862.final.vcf.gz ./SRR6023863.final.vcf.gz
./SRR6023864.final.vcf.gz ./SRR6023865.final.vcf.gz ./SRR6023882.final.vcf.gz
./SRR6023883.final.vcf.gz ./SRR6023884.final.vcf.gz ./SRR6023885.final.vcf.gz >
merge.vcf
```

```
grep "SRR6023853" merge.vcf
grep "SRR6023854" merge.vcf

#上述两个命令是否都可以得到输出信息
```

如此可以得到正确的合并文件merge.vcf

对vcf文件进行压缩的时候要使用bgzip，这是vcf文件的压缩格式：

```
conda install tabix
bgzip ./*.vcf

tabix -p vcf SRR6023853.final.vcf #构建索引
```

之后得到的vcf将其转换为phy格式,这样的文件就包含所有的样本了：

```
nohup python vcf2phylip.py -
-/user/WH/workspace/train_data_test/2.callsnp_gatk/4.filter_opteration/5.final_fi
lter_SNP_and_cat/vcf_sample/merge.vcf > vcf2phy.log 2>&1 &
```

得到的merge.min4.phy进行建树：

```
iqtree -s ./merge.min4.phy -nt 15 -m MF -st DNA -o Europe
```

上述命令可以直接计算出最适合的替代模型，iqtree会尝试536种模型并得到最适的，这一过程耗时18h左右

最后我们可以得到一个merge.min4.treefile，将其放入Figtree中即可观察到最终的进化树。

```
/user/WH/workspace/train_data_test/2.callsnp_gatk/4.filter_opteration/5.final_fil
ter_SNP_and_cat/vcf_sample/merge.vcf
```

实验记录 PCA

PCA步骤

```
conda install plink
```

将vcf转换为plink文件,注意这里要使用merge.vcf

```
nohup plink --vcf
/user/WH/workspace/train_data_test/2.callsnp_gatk/4.filter_opteration/5.final_fil
ter_SNP_and_cat/vcf_sample/merge.vcf --recode --out testacc --const-fid --allow-
extra-chr > trans2plink.log 2>&1 &

# --vcf vcf 或者vcf.gz
# --recode 输出格式
# --out 输入前缀
# --const-fid  添加群体信息
# --allow-extra-chr 允许非标准染色体编号
```

```
nohup plink --allow-extra-chr --file testacc --noweb --make-bed --out testacc >
bed.log 2>&1 &

# --file .ped + .map 文件前缀
# --make-bed 建立一个新的二进制文件
```

```
nohup plink --allow-extra-chr --threads 20 -bfile testacc --pca 2 --out testacc >
pca.log 2>&1 &


# --threads 线程数
# --pca 主成分
```

得到2个以.eigenval, .eigenvec结尾的文件;其中.eigenval代表每个PCA所占的比重，另外一个记录特征向量，用于坐标轴的绘制

绘制了pca=2和pca=3的结果

这里要进行可视化，需要先生成一个表格，这将用到我们之前构建的进化树的结果

选择树的每一分支长度最为合适的作为一个假设，以此作为群体信息（因为没有多余的信息了？）

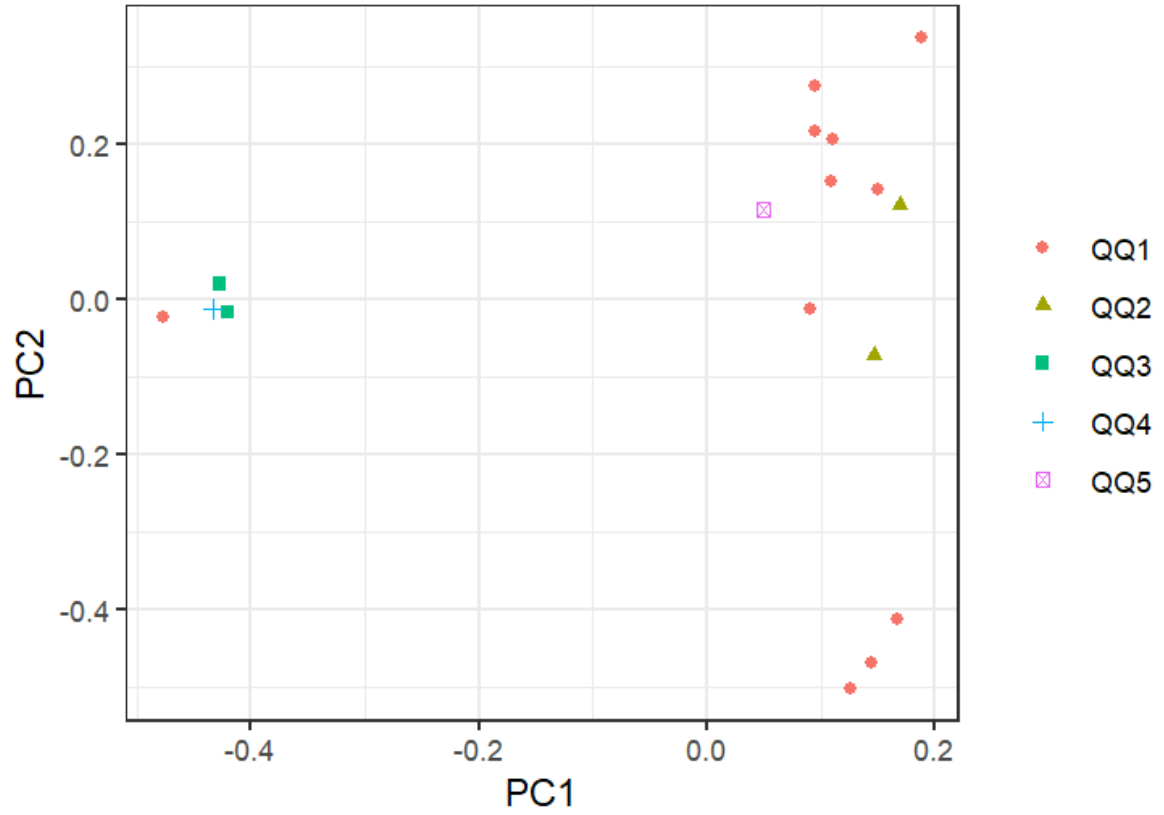
```
pop name    pca1    pca2
QQ1 tSM_SRR6023885  -0.476561   -0.0234431
QQ1 tSM_SRR6023864  0.150183    0.142417
QQ1 tSM_SRR6023859  0.09505 0.276106
QQ1 tSM_SRR6023861  0.090877    -0.0132192
QQ1 tSM_SRR6023858  0.10962 0.151507
QQ1 tSM_SRR6023860  0.126306    -0.502292
QQ1 tSM_SRR6023856  0.167568    -0.412636
QQ1 tSM_SRR6023862  0.0944507   0.217427
QQ1 tSM_SRR6023854  0.144114    -0.468221
QQ1 tSM_SRR6023853  0.187758    0.337843
```

```
QQ1 tSM_SRR6023882  0.110328    0.20584
QQ2 tSM_SRR6023883  0.146754    -0.0722683
QQ2 tSM_SRR6023865  0.169564    0.12147
QQ3 tSM_SRR6023855  -0.427262   0.0199812
QQ3 tSM_SRR6023863  -0.419949   -0.0157297
QQ4 tSM_SRR6023857  -0.433819   -0.0123061
QQ5 tSM_SRR6023884  0.0498027   0.115973
```

之后使用ggplot2进行绘图：

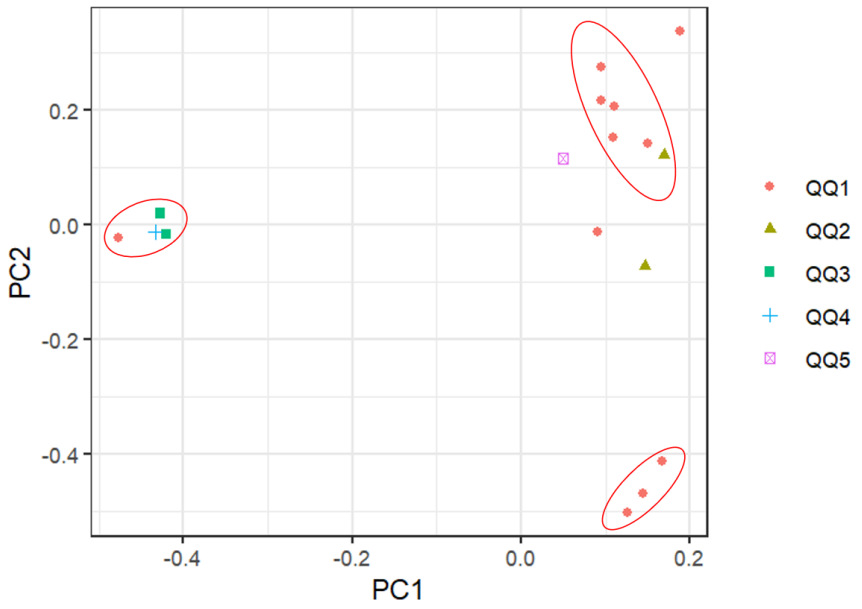
```
(ggplot2)
pca <- read.table("D:/pop.txt",sep = "\t",header = T,fileEncoding="UTF16")
library(ggplot2)
ggplot(pca, aes(x=pca1,y=pca2)) +
  geom_point(aes(color=pop, shape=pop),size=1.5)+
  labs(x="PC1",y="PC2")+theme_bw()+theme(legend.title = element_blank())
```

得到的结果如下：



上图显示在PC1水平上，QQ1样本和QQ2样本、QQ5样本无显著差异，前三者和QQ3、QQ4有显著差异

在PC2水平上，QQ1内的水平发生了分化


0.2
0.0
-0.2
-0.4
PC2
-0.4
-0.2
0.0
0.2
PC1
QQ1
QQ2
QQ3
QQ4
QQ5

实验记录 使用GATK4进行SNPs和indels的挖掘

本实验的raw data 是编号为**SSR6023853-SSR6023865,SSR6023882-SSR6023885的fastq格式**数据，测序长度为PE150,参考基因组为**my.ref.fa**,测序的方式是RNA-Seq。

通过前面的过滤raw data和比对操作，我们可以得每一组测序数据与reference比对生成的sam文件，将所有的sam文件转为bam文件之后，首先是对每一个文件进行sort，之后执行merge操作（merge之后的bam也是排序好的，理论上是不需要sort的，但我这里又sort了一遍），此时我们可以得到数量为1的bam文件，它是一个包含了我们所有测序数据的比对结果的二进制文件，之后进行GATK的相关操作来进行变异的挖掘。

```
samtools sort -@ 10 merged.bam -o merged_sort.bam
```

工具的下载

picard:下载最新版本的Picard Tools - By Broad Institute

gatk4:下载最新版本的Releases · broadinstitute/gatk (github.com)

java(最新版的)：在官网下载最新版本就行，注意一定要下x64版本的，实验室服务器的CPU是intel的不是ARM架构的

数据的清理

标记PCR重复

最新版本的GATK将picard已经纳入其中了，所以我们直接使用GATK命令就可以了

```
$raw_bam = ./merged.bam
$java =  #这里设置环境变量之后可以不填
$gatk = #这里设置环境变量之后可以不填
nohup $java $gatk MarkDuplicates -I $raw_bam -O ./merged_markdup.bam -M
metrics.txt   --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP 1000 > gatk_mkdup.log 2>&1 & #
这一步时间会很长
```

运行会生成`merged_bam_markdup.bam`和`metrics.txt`

-M metrics.txt设置输出报告，其中包含有关重复标记过程的统计信息。

标记完成之后就可以构建索引文件了：

使用GATK callSNPs需要的索引文件有三个：**reference的.fai、reference的.dict、merged_bam_markdup.bam文件的.bai文件。**

对bam构建idx

这一步要在去重之后进行，否则会增加算力负担（最好在全部处理完成得到最终bam之后进行）

```
samtools index merged_markdup.bam
```

对reference构建idx

```
$ref=./my.ref.fa
$java $gatk CreateSequenceDictionary -R $ref -O my.ref.dict
samtools faidx $ref
```

对indel附近进行重新比对

realignment操作

要使用GATK对indel周围进行重新比对，你可以使用GATK工具中的`IndelRealigner`。这个工具可以根据已知的indel信息重新对reads进行比对，以改善indel区域的比对质量。

创建一个目标坐标数据文件（`target intervals`）：

```
gatk RealignerTargetCreator -R $ref -I merged_markdup.bam -o target.intervals
```

这个命令会分析去重的BAM文件（`merged_bam_markdup.bam`），识别其中的indel位置，并生成一个目标坐标数据文件（`target intervals`）。

对reads进行重新比对：

```
gatk IndelRealigner -R $ref -I merged_markdup.bam -targetIntervals
target.intervals -o merged_markdup_realigned.bam
```

这个命令会根据目标坐标数据文件（`target intervals`）对reads进行重新比对，并输出一个重新比对后的BAM文件（`merged_markdup_realigned.bam`）。

可选：如果你想生成一个基于实际比对观察到的indel的索引文件，以用于以后的分析步骤，你可以执行以下命令：

```
gatk BamIndexGenerator -I realigned.bam
```

这个命令会生成一个索引文件，它可以为实际比对观察到的indel提供快速检索。

请注意，上述命令中的参数和文件名可能需要根据你的具体情况进行调整。尤其是`-R`参数要指定正确的参考基因组文件，`-I`参数要指定正确的输入BAM文件，以及`-o`参数要指定输出文件的名称。

碱基质量矫正Base quality score recalibration (BQSR)

```
gatk BaseRecalibrator -I merged_markdup_realigned.bam -R $ref --known-sites
dbSNP.vcf.gz -O recal_data.table
```

`dbSNP.vcf`是http://www.ncbi.nlm.nih.gov/SNP/NCBI的公开数据，我们做的植物如果没有的话可以不设置这个参数，不会影响最终结果。

生成的`recal_data.table`其中记录了参考基因组与样本数据之间的差异，以及需要校正的碱基质量信息。

```
#将BQSR结果进行应用，生成最终的bam文件
gatk ApplyBQSR -R $ref -I merged_bam_markdup.bam --bqsr-recal-file
recal_data.table -O merged_markdup_realigned_recal.bam
```

最终过滤完成的文件为`merged_markdup_realigned_recal.bam`

使用GATK进行Variant Calling

最新的GATK版本给出的算法为`HaplotypeCaller`：

```
gatk HaplotypeCaller -R $ref -I merged_markdup_realigned_recal.bam --dbsnp
dbSNP.vcf.gz -O raw_variants.vcf
```

最后得到`raw_variants.vcf`，`--dbsnp dbSNP.vcf.gz`这一项没有公共数据也可以不写。

进行VCF文件的过滤

在使用GATK进行变异位点调用后，通常需要对原始的VCF文件进行进一步过滤，以过滤掉可能的假阳性（false positive）变异。

下面是使用GATK提供的工具VariantFiltration来对VCF文件进行过滤的一般步骤：

1. 创建一个过滤条件（Filter Expression）：根据你的需求，创建一个过滤条件来过滤掉低质量的变异位点。例如，你可以将一些常见的过滤条件包括：QUAL < 30.0 (质量值低于30)，DP < 10 (reads深度低于10)，FS > 60.0 (Fisher Strand值大于60)等等。你可以根据具体的需求和实验数据来自定义这些过滤条件。
2. 使用`VariantFiltration`进行过滤：运行`VariantFiltration`工具，并应用上述定义的过滤条件来过滤VCF文件。

```
gatk VariantFiltration -R $ref -V raw_variants.vcf --filter-expression "过滤条件"
--filter-name "过滤名称" -O filtered_variants.vcf
```

其中，`$ref`是参考基因组的FASTA文件路径，`raw_variants.vcf`是HaplotypeCaller生成的原始VCF文件，`过滤条件`是你定义的过滤条件，`过滤名称`是为过滤条件定义的名称，`filtered_variants.vcf`是输出的过滤后的VCF文件。

3. 可选：可以根据你的需要添加或修改过滤条件，并多次运行VariantFiltration命令来进一步优化过滤参数和过滤结果。

4月12日组会

目前的工作方向

FST和Π要看一下怎么算，也就是群体间分歧度检验和核酸多样性

好好看下树，树很重要，如果树很明确的话后面两部也很简单了，会对上的

文献学习

Tracing the geographic origin of endangered plant species using transcriptome-derived SNPs: An example of Cathaya argyrophylla

本文提出了一种基于转录组数据挖掘SNP位点以追溯濒危物种的地理起源的方法

这篇文章和我做的有类似之处，都是研究一种物种的群体结构的，很有参考价值

摘要

提到了银杉树种是在中国的四个山区内才有的树种，然后采样是**覆盖了全部区域的五个群体**，说明这个采样应该是一种抽样策略

结论：

最后算出的核酸多样性很低，但群体特异性很高

鉴定出了群体特异性SNP位点，以此为依据赘述物种地理起源，并设计了双盲实验进行验证

流程开发日志——青海云杉

这是一个实验，因为之前组里用的脚本是用bwa比对的，老师要求以后都用hisat，因而我改变了比对软件，并以学习的目的自己写了整套代码，没有用组里的脚本，所以为了验证我的做法是不是正确的，所以用这个流程算一批已经算过的数据，看结果是不是一样或者合理的

数据获取

数据是从师姐的硬盘里转存的，要用md5检验一下完整性

```
python 01.MD5.py #生成01.MD5.sh

chmod u+x 01.MD5.sh #添加可执行权限

./01.MD5.sh > CHECK.log #如果是确定就没有问题
```

过滤

```
cd = "/user/WH/P.crassifolia"
list=[]
current='.'
for folder in os.listdir(cd):
    if os.path.isdir(os.path.join(cd, folder)):
        list.append(folder)
```

从下载的文件夹中可以提取出每个sample命名的base name，这样生成一个list之后可以很方便我们写接下来的脚本(现在总结出的一个技巧，虽然每次跑流程都要重新写脚本，但是可以很大程度提高代码的复用率)

fastp命令设置：

```
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i /user/WH/P.crassifolia/XQ2-1-ZL/XQ2-1-ZL_1.fq.gz -I /user/WH/P.crassifolia/XQ2-1-ZL/XQ2-1-ZL_2.fq.gz -o ./XQ2-1-ZL_1_filt.fq.gz -O ./XQ2-1-ZL_2_filt.fq.gz
```

运行fastp.py生成01.fastp.sh

结果文件：

```
/user/WH/workspace/P.crassifolia_0304/01.fastp/clean_data
```

call SNP

mapping

使用hisat2工具进行比对：

对参考基因组构建索引：

```
hisat2-build -p 4 genome.fa genome #这里可以直接用参考基因组的名字就行 my.ref.fa
```

genome.fa就是my.ref.fa,随后进行比对和索引排序：

```
hisat2 -p 15 -x
/user/WH/workspace/train_data_test/1.callSNP_samtools/2.mapping/reference/genome
-1 /user/WH/workspace/P.crassifolia_0304/01.fastp/clean_data/XQ2-1-
ZL_1_filt.fq.gz -2 /user/WH/workspace/P.crassifolia_0304/01.fastp/clean_data/XQ2-
1-ZL_2_filt.fq.gz -S ./sam/XQ2-1-ZL.sam &&

samtools view -F 4 -bS ./sam/XQ2-1-ZL.sam -o ./sam2bam/XQ2-1-ZL.bam &&

samtools sort -@ 15 ./sam2bam/XQ2-1-ZL.bam -o ./sort/XQ2-1-ZL.sorted.bam &&

samtools index ./sort/XQ2-1-ZL.sorted.bam
```

得到最终的bam文件

标记PCR重复

qPCR测序会由于扩增差异导致DNA的重复序列数量不同，这样会影响我们后续对SNP数量的统计，因此要进行标记PCR重复操作。

操作的原理是gatk利用SAM文件中的信息确定比对序列的来源情况，进而判断其是否为重复序列(Mark duplicates)

```
gatk MarkDuplicates -I /user/WH/workspace/P.crassifolia_0304/02.hisat2/sort/XQ2-
1-ZL.sorted.bam -O ./XQ2-1-ZL_Markdup.bam -M metrics.txt --
MAX_FILE_HANDLES_FOR_READ_ENDS_MAP 1000 --REMOVE_DUPLICATES true
```

其中`--REMOVE_DUPLICATES true`设置直接去除duplicate，这一步要设置，可以减少计算量增加速度

添加Read Group

GATK目前不支持没有Read Group的sam文件，因为后续进化树的构建需要用到RG的信息，所以这里使用picard进行操作

```
picard AddOrReplaceReadGroups --INPUT
/user/WH/workspace/P.crassifolia_0304/03.markduplicates/data/XQ2-1-ZL_Markdup.bam
--OUTPUT ./XQ2-1-ZL_ADD.bam --RGLB tLB_XQ2-1-ZL --RGPL ILLUMINA --RGPU unit1 --
RGSM XQ2-1-ZL
```

-`--RGSM XQ2-1-ZL`这里就填你样品的名字最好，因为后面的进化树也会显示一样的tip，也可以后面用figtree批量修改，之后还要构建索引

```
samtools view -H BS-1-ZL_ADD.bam | grep '@RG' #检验是否添加成功
```

indel realignment

首先要对文件进行indel信息的标记，生成.interval文件

```
gatk3 -nt 15 -R /user/WH/Picea_wilsonii_Mast/reference/my.ref.fa -T
RealignerTargetCreator -o ./XQ2-1-ZL.realn.intervals -I
/user/WH/workspace/P.crassifolia_0304/04.add_read_group/ADD/XQ2-1-ZL_ADD.bam --
filter_reads_with_N_cigar
```

再进行indel的重新比对，如果没有这一步的话vcf文件会出错，之前的表现是所有的ALT碱基都是*号

```
gatk3 -R /user/WH/Picea_wilsonii_Mast/reference/my.ref.fa -T IndelRealigner --
targetIntervals ./XQ2-1-ZL.realn.intervals -o ./XQ2-1-ZL.realn.bam -I
/user/WH/workspace/P.crassifolia_0304/04.add_read_group/ADD/XQ2-1-ZL_ADD.bam --
filter_reads_with_N_cigar
```

如果使用的gatk4，则需要先wget package，再设置最新版本的java环境：

```
export JAVA_HOME=/user/WH/software/jdk-21.0.1 #这里是wget的目录
export PATH=$JAVA_HOME/bin:$PATH
source ~/.bashrc
```

并且在参数的书写格式上有些区别，可以自行摸索，结果是一样的

2024-0308 这次跑数据报错了以下内容：

```
ERROR MESSAGE: An error occurred because you did not provide enough memory to run
this program. You can use the -Xmx argument (before the -jar argument) to adjust
the maximum heap size provided to Java. Note that this is a JVM argument, not a
GATK argument.
```

是因为内存出错了，这里我手动设置了最大堆内存为30G（这里其实设置高了，10g就够了）：

```
nohup java -Xmx30g -jar /user/WH/miniconda3/pkgs/gatk-3.8-hdfd78af_11/opt/gatk-
3.8/GenomeAnalysisTK.jar -R /user/WH/Picea_wilsonii_Mast/reference/my.ref.fa -T
IndelRealigner --targetIntervals ./HL01-1.realn.intervals -o ./HL01-1.realn.bam -
I /user/WH/workspace/P.crassifolia_0304/04.add_read_group/ADD/HL01-1_ADD.bam --
filter_reads_with_N_cigar > test.log 2>&1 &
```

值得知道的是现在网上已经下不到gatk3的jar包了，因而只能用conda安装，而这需要我们按以下路径来找到jar的位置：

```
miniconda3/pkgs/gatk-3.8-hdfd78af_11/opt/gatk-3.8/GenomeAnalysisTK.jar
```

call SNP

```
samtools mpileup -g -t DP,DP4,SP -f
/user/WH/Picea_wilsonii_Mast/reference/my.ref.fa -o ./XQ2-1-ZL.bcf
/user/WH/workspace/P.crassifolia_0304/05.indel_realignment/XQ2-1-ZL.realn.bam &&
bcftools call -cvM -o ./XQ2-1-ZL.vcf
/user/WH/workspace/P.crassifolia_0304/06.call_SNP/bcf/XQ2-1-ZL.bcf
```

生成的文件为raw vcf，需要过滤

SNP过滤+合并+过滤

先进行vcf文件的压缩：

```
gzip *.vcf
```

使用 `filter_indel.pl` 过滤indel附近5bp的数据，再使用 `filter_dp_qual.pl` 过滤深度和质量值

```
perl filter_dp_qual.pl
/user/WH/workspace/P.crassifolia_0304/07.filter_operation/filt_indel/XQ2-1-
ZL_filt1.vcf.gz ./XQ2-1-ZL_filt2.vcf.gz
perl filter_indel.pl
/user/WH/workspace/P.crassifolia_0304/06.call_SNP/raw_vcf/XQ2-1-ZL.vcf.gz ./XQ2-
1-ZL_filt1.vcf.gz
```

之后得到的vcf要进行解压缩，并且使用另外的一种方式进行压缩和构建索引：

```
for file in ./*.vcf; do
    if [ -e "$file" ]; then
        bgzip "$file"
        tabix -p vcf "${file}.gz"
    fi
done
```

再对vcf文件进行合并，得到merge.vcf

```
bcftools merge BS-1-ZL_filt2.vcf.gz BS2-1-ZL_filt2.vcf.gz HL01-1_filt2.vcf.gz
HL03-1_filt2.vcf.gz HL04-1_filt2.vcf.gz HL05-3_filt2.vcf.gz HL07-2_filt2.vcf.gz
HX-1-ZL_filt2.vcf.gz HZ-1-ZL_filt2.vcf.gz HZ2-1-ZL_filt2.vcf.gz LD2-1-
ZL_filt2.vcf.gz LD-2-ZL_filt2.vcf.gz MT-1-ZL_filt2.vcf.gz QL01-3_filt2.vcf.gz
QL02-2_filt2.vcf.gz QL-1-ZL_filt2.vcf.gz QL2-1-ZL_filt2.vcf.gz SD-1-
ZL_filt2.vcf.gz SN-1-ZL_filt2.vcf.gz SN2-1-ZL_filt2.vcf.gz TZ-1-ZL_filt2.vcf.gz
XM-1-ZL_filt2.vcf.gz XQ2-1-ZL_filt2.vcf.gz > merge.vcf.gz
```

之后再进行过滤：

```
vcftools --gzvcf ./merge.vcf.gz  --minDP 10 --max-missing-count 5 --recode --
recode-INFO-all --stdout | gzip -c > merge.dp10ms5.vcf.gz &&
vcftools --gzvcf ./merge.dp10ms5.vcf.gz --min-alleles 2 --max-alleles 2 --non-
ref-af 0.01 --recode --recode-INFO-all --stdout | gzip -c >
merge.dp10ms5.snp.alte-2.vcf.gz

--max-missing-count 5设置是23*0.2得来的
```

设置了最小测序深度（`--minDP 10`）和最大缺失计数（`--max-missing-count 5`），然后使用 `--recode` 选项将过滤后的VCF数据重编码为一种更简洁的格式，并通过 `--stdout` 将输出发送到标准输出，并通过管道传递给 `gzip -c` 进行压缩，最终生成 `merge.dp10ms5.vcf.gz` 文件，生成的 `merge.dp10ms5.vcf.gz` 文件进行进一步处理，设置了最小等位基因数（`--min-alleles 2`）和最大等位基因数（`--max-alleles 2`），以及非参考等位基因频率（`--non-ref-af 0.01`），再次使用 `--recode` 和 `--recode-INFO-all` 进行重编码，并通过管道传递给 `gzip -c` 进行压缩，最终生成 `merge.dp10ms5.snp.alte-2.vcf.gz` 文件

vcf2fasta

使用`vcf2fa.pl`脚本进行转换，转换后的fasta除了样本之外，还有欧洲云杉的外类群：

```
perl vcf2fa.pl merge.dp10ms5.snp.alte-2.vcf.gz 23 > merge.dp10ms5.snp.alte-
2.vcf.fa
```

最终得到的VCF文件包含所有个体的变异信息，我们需要用下面这个脚本把其转换为fasta文件：

```
#!/usr/bin/perl
use strict;
use warnings;
# 从命令行参数读取文件路径和样本数量
my $file = $ARGV[0];
my $sampleNO = $ARGV[1];
# 打开文件，使用 zcat 来解压.gz文件，如果文件是压缩的
open(FILE, "zcat $file |");
# 初始化变量，存储样本名称、基因型数据和染色体比较顺序
my (@sampleName, %hash, %comp);
# 计数器，用于染色体的比较顺序
my $count = 0;
# 杂合子基因型转换表
my %het = ('AC' => 'M', 'AG' => 'R', 'AT' => 'W', 'CA' => 'M', 'CG' => 'S', 'CT'
=> 'Y',
           'GA' => 'R', 'GC' => 'S', 'GT' => 'K', 'TA' => 'W', 'TC' => 'Y', 'TG'
=> 'K');
# 逐行读取文件内容
while (<FILE>) {
    # 跳过以 ## 开头的注释行
    next if (/##/);
    # 如果是列标题行，提取样本名称并添加到样本名称数组
    if (/^#CHROM/) {
        for my $i (0 .. $sampleNO - 1) {
            push(@sampleName, (split(/\s+/))[9 + $i]);
        }
        push(@sampleName, "Europe");  # 添加 "Europe" 样本
        next;
    }
    # 提取染色体名、参考序列、替代序列和位置
    my $chrom = (split(/\s+/))[0];
    my $ref = (split(/\s+/))[3];
    my $alt = (split(/\s+/))[4];
    my $pos = (split(/\s+/))[1];
    # 如果染色体不在比较顺序哈希中，则添加并更新计数器
    unless (exists $comp{$chrom}) {
        $count++;
        $comp{$chrom} = $count;
    }
    # 遍历每个样本，解析基因型数据
    for my $j (0 .. $sampleNO - 1) {
        my $var = (split(/\s+/))[9 + $j];
        my $genotype = (split(/:/, $var))[0]; #提取基因型的信息
        my @fields = split(/\//, $genotype);
        # 根据基因型的不同情况，存储数据到 hash 中
```

```
        if ($fields[0] eq '.' && $fields[1] eq '.') {
            $hash{$chrom}{$pos}{$sampleName[$j]} = "N";
        } elsif ($fields[0] eq '0' && $fields[1] eq '0') {
            $hash{$chrom}{$pos}{$sampleName[$j]} = $ref;
        } elsif ($fields[0] eq '0' && $fields[1] eq '1') {
            $hash{$chrom}{$pos}{$sampleName[$j]} = $het{"$ref$alt"};
        } elsif ($fields[0] eq '1' && $fields[1] eq '1') {
            $hash{$chrom}{$pos}{$sampleName[$j]} = $alt;
        }
    }
    # 为 "Europe" 样本添加参考序列
    $hash{$chrom}{$pos}{Europe} = $ref;
}
# 关闭文件
close(FILE);
# 打印每个样本的基因型数据
foreach my $name (@sampleName) {
    print ">$name\n";
    foreach my $chrom (sort {$comp{$a} <=> $comp{$b}} keys %comp) {
        foreach my $pos (sort {$a <=> $b} keys %{$hash{$chrom}}) {
            print $hash{$chrom}{$pos}{$name};
        }
    }
    print "\n";
}
```

整个脚本的编写顺序和人阅读的顺序是一样的，其基本逻辑就是从VCF文件的基因型信息（genotype）中挖掘出具体的位点信息

建立系统发育树

这里有几个问题都遇到了：

首先是在merge之后没有进行过滤，，或者过滤的参数--max-missing-count设置的过高，这一步将导致fasta文件中有很多N,以及外类群序列和个体序列长度相差过大，导致iqtree无法运行：

```
ERROR: Sequence BS2-1-ZL contains not enough characters (472643)
ERROR: Sequence HL01-1 contains not enough characters (472560)
ERROR: Sequence HL03-1 contains not enough characters (472402)
ERROR: Sequence HL04-1 contains not enough characters (472479)
ERROR: Sequence HL05-3 contains not enough characters (472380)
ERROR: Sequence HL07-2 contains not enough characters (472410)
ERROR: Sequence HX-1-ZL contains not enough characters (472503)
ERROR: Sequence HZ-1-ZL contains not enough characters (472518)
ERROR: Sequence HZ2-1-ZL contains not enough characters (472531)
ERROR: Sequence LD2-1-ZL contains not enough characters (472533)
ERROR: Sequence LD-2-ZL contains not enough characters (472344)
ERROR: Sequence MT-1-ZL contains not enough characters (472600)
ERROR: Sequence QL01-3 contains not enough characters (472483)
ERROR: Sequence QL02-2 contains not enough characters (472413)
ERROR: Sequence QL-1-ZL contains not enough characters (472470)
ERROR: Sequence QL2-1-ZL contains not enough characters (472532)
ERROR: Sequence SD-1-ZL contains not enough characters (472551)
ERROR: Sequence SN-1-ZL contains not enough characters (472411)
ERROR: Sequence SN2-1-ZL contains not enough characters (472450)
```

```
ERROR: Sequence TZ-1-ZL contains not enough characters (472559)
ERROR: Sequence XM-1-ZL contains not enough characters (472521)
ERROR: Sequence XQ2-1-ZL contains not enough characters (472266)
ERROR: Sequence Europe contains too many characters (472815)
```

其次是在iqtree的参数设置的时候：

```
iqtree -s merge.dp10ms5.snp.alte-2.vcf.fa -nt 40 -m MF
```

这一命令有两个问题，第一个是建议使用-m MFP比较好，第二个是-st DNA没有设置，这将导致运算过长，第三是没有指定外类群-o Europe

```
iqtree -s merge.dp10ms5.snp.alte-2.vcf.fa -nt 40 -m MFP -o Europe
```

这里的外群即参考基因组，通过前面的脚本我们在fasta文件中构建了这一个体，命名其为Europe

还有一个问题就是,文件中出现了逗号导致软件无法识别。可以使用以下命令消除：

```
sed 's/,//g' | merge.fa
```

最终使用iqtree发现分群效果不对，显示欧洲云杉的亲缘关系过于远了，考虑忽略枝长即可

最合适的命令如下：

```
nohup iqtree -s picea_asperata.fa -nt 40 -m MFP -o Europe -alrt 1000 -bb 1000 -st
DNA > iqtree.log 2>&1 &
```

主成分分析

```
$vcftools --gzvcf $file --plink --out pca_$basefile/$basefile.plink
```

把vcf文件转换为plink文件,会生成map文件

```
"sed -i 's/^0/1/' pca_$basefile/$basefile.plink.map\n"
```

将map文件的染色体0全部变成1

map格式的文件, 主要是图谱文件信息, 主要包括染色体名称, 所在的染色体和所在染色体的坐标.
map文件包括：

- 染色体编号（1-22, X, Y or 0 if unplaced）, 未知为0
- SNP名称（字符或数字）, 如果不重要, 可以从1编号, 注意要和bed文件SNP列一一对应
- 染色体的摩尔位置(可选项, 可以用0)
- SNP物理坐标

```
$plink --file pca_$basefile/$basefile.plink --indep-pairwise 50 5 0.2 --out
pca_$basefile/$basefile.plink --noweb
```

这一步是为了标记连锁不平衡现象

```
$plink --file pca_$basefile/$basefile.plink --extract
pca_$basefile/$basefile.plink.prune.in --make-bed --out
pca_$basefile/$basefile.plink.pruneddata --noweb
```

这些命令产生一个修剪过的SNP子集，这些标记之间大致处于连锁平衡状态，这些SNP的id被写入plink.prune.in，并且所有被排除的SNP的id被写入plink.prune.out。

```
$plink --bfile pca_$basefile/$basefile.plink.pruneddata --recode --out
pca_$basefile/$basefile.plink.pruneddata --noweb #输出正常格式

$plink --file pca_$basefile/$basefile.plink.pruneddata --recode12 --out
pca_$basefile/$basefile.plink.pruneddata.recode12 --noweb #将次等位基因变为1，主等位
基因变为2
```

安装EIGENSOFT：

```
conda install EIGENSOFT

export PATH=/user/WH/software/EIG-6.1.4/bin:\$PATH

#进行PCA分析
smartpca.perl -i pca_$basefile/$basefile.plink.pruneddata.recode12.ped \\
              -a pca_$basefile/$basefile.plink.pruneddata.recode12.map \\
              -b pca_$basefile/$basefile.plink.pruneddata.recode12.pedind \\
              -o pca_$basefile/$basefile.plink.pruneddata.recode12.PCA \\
              -p pca_$basefile/$basefile.plink.pruneddata.recode12.PCA.plot \\
              -e
pca_$basefile/$basefile.plink.pruneddata.recode12.PCA.eigenvalues \\
              -l pca_$basefile/$basefile.plink.pruneddata.recode12.PCA.log
```

可视化：

```
perl ggplot.pl pca_$basefile/$basefile.plink.pruneddata.recode12.PCA.evec
$Rscript pca_$basefile/$basefile.plink.pruneddata.recode12.PCA.evec.R
```

```
$plink --file pca_$basefile/$basefile.plink --recode12 --out
pca_$basefile/$basefile.plink.recode12 --noweb
#这一步是将原始的数据进行recode12，方便我们进行下一步的admixture
```

```
#ggplot.pl

my $f=shift;
my $out="$f.ggplot2";
open(I,"< $f");
<I>;
my @head=("FID");

for(my $i=1;$i<=10;$i++){
    my $h=PC."$i";
    push @head,$h;
}
push @head,"species";
```

```
my $head=join "\t",@head;
open(O,"> $out");
print O "$head\n";
while(<I>){
    chomp;
    s/^\s+//;
    print O "$_\n";
}
close I;
close O;

open(R,"> $f.R");
print R "
.libPaths(\"\/data\/00\/user\/user108\/software\/R\/x86_64-redhat-linux-gnu-
library\/4.0\");
library(\"ggplot2\");
a=read.table(\"$f.ggplot2\",header=T);
pdf(file=\"$f.PC1_PC2.pdf\");
ggplot(a,aes(PC1,PC2,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
pdf(file=\"$f.PC1_PC3.pdf\");
ggplot(a,aes(PC1,PC3,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
pdf(file=\"$f.PC1_PC4.pdf\");
ggplot(a,aes(PC1,PC4,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
pdf(file=\"$f.PC2_PC3.pdf\");
ggplot(a,aes(PC2,PC3,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
pdf(file=\"$f.PC2_PC4.pdf\");
ggplot(a,aes(PC2,PC4,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
pdf(file=\"$f.PC3_PC4.pdf\");
ggplot(a,aes(PC3,PC4,color=species))+geom_point(alpha=0.5,size=2);
dev.off;
";
close R;
`/data/apps/R-3.4.0/bin/Rscript $f.R`;
```

群体结构分析

使用admixture工具：

对prune的数据和raw数据都进行admixture：

```
for (my $i=1;$i<=23;$i++) {
                    print "$admixture -j16 --cv=10 $ped $i
>$basename.log$i.out\n";
                }
```

选择了K取1-23算得CV error最小为K=22的时候(有点夸张,说明这个数据的混杂程度有点高，处于基因交流的中心地带)

TIPs: 后面改了个体数，从23个缩减到了11个，主要是判断树形差不多就行了，然后PCA的结果不好，聚的不够紧，但是这里群体抽样了所以可以理解，验证的依据主要是看树形，最后的树形还算合理

```
#总的脚本是
use strict;
use warnings;

my
$dir="/user/WH/workspace/P.crassifolia_0304/09.PCA2/pca_merge.dp10ms5.snp.alte-
2.vcf.gz";
my @files=glob($dir);

my $admixture="~/miniconda3/bin/admixture";
foreach my $file (@files) {
    chomp $file;
    if(-d $file){
        my @ped=`ls $file/*.ped`;
        foreach my $ped (@ped) {
            chomp $ped;
            if($ped=~/recode12\.ped/){
                my $basename=`basename $ped`;
                chomp $basename;
                print "mkdir $basename\n";
                print "cd $basename\n";
                for (my $i=1;$i<=10;$i++) {
                    print "$admixture -j16 --cv=10 $ped $i
>$basename.log$i.out\n";
                }
                print "cd ..\n";
                print "cd $basename;perl ../admixture.plot.pl $ped;cd ..\n";
            }
            print "\n\n";
        }
    }
}
print "perl plot.pl\n";
```

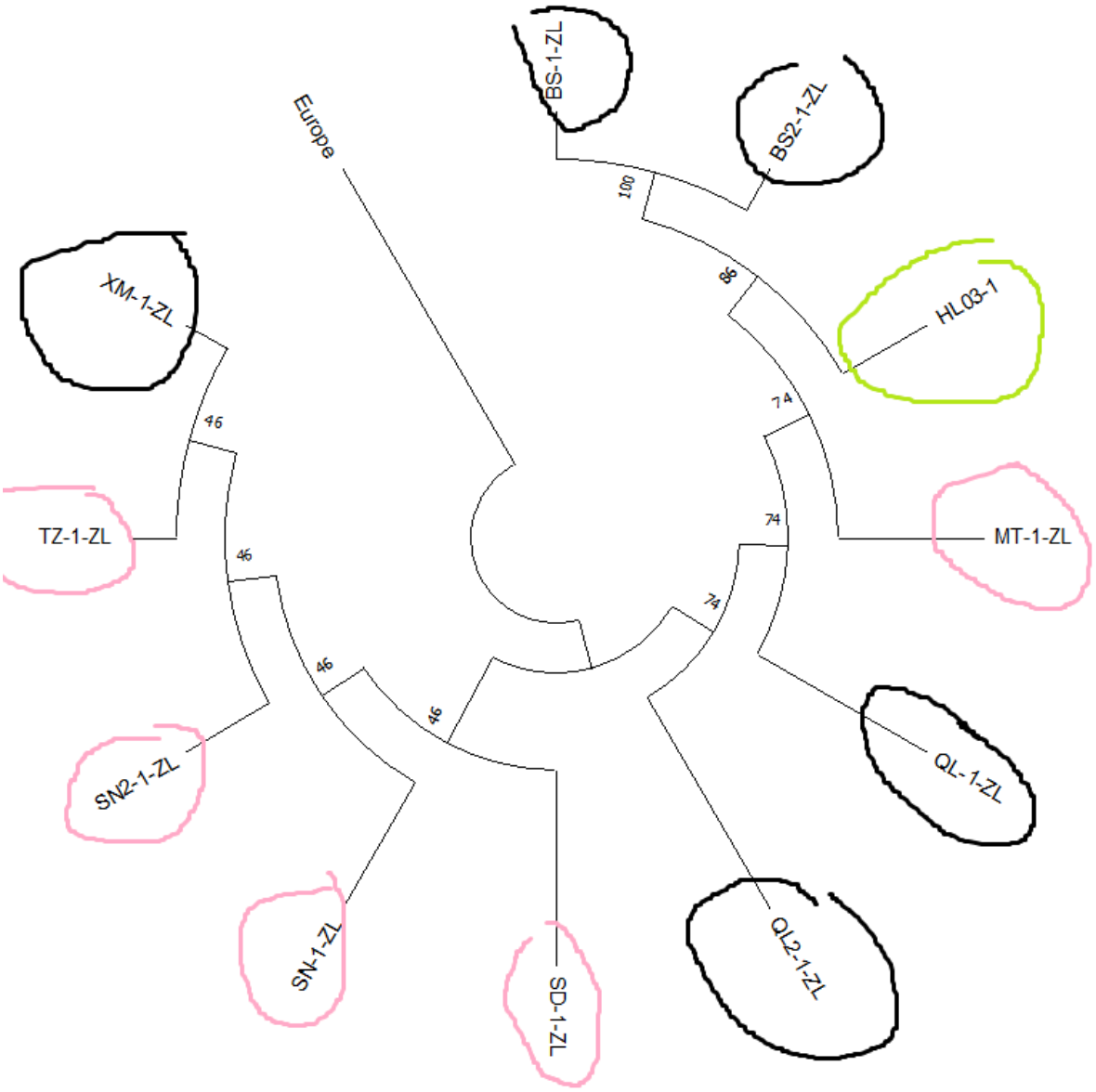
可视化:

```
my @file=`ls *.Q`;


my %hash1;
my %pop;
my $pedfile=shift;
my $list="/user/WH/workspace/P.crassifolia_0304/10.Admixture4/admixture.txt";
###jueduilujing

open(R,'>',"runplot.R");
print R "
.libPaths(\"\/data\/00\/user\/user108\/software\/R\/x86_64-redhat-linux-gnu-
library\/4.0\");
library(\"ggplot2\");library(\"grid\")\n";

my @pop=&readPop($list);
foreach my $file(@file){
```

```
    my %hash;
    chomp $file;
    my $ped;
    my $prefix;
    my $suffix;
    if($file=~m/^(\S+)\.(\d+).Q/){
    $prefix="$1";
    $suffix=$2;
    $ped=$pedfile;
    }else{
    die "$file\n";
    }
    my @ind=&readPed($ped);
    my $out1="$prefix.$suffix.result";
    my $out2="$prefix.$suffix.result.ggplot";
    open(O1,'>',"$out1");
    open(O2,'>',"$out2");
    open(F,$file);
    print O2 "id\tpercent\tk\n";
    my $i=0;
    while(<F>){
    chomp;
    my @a=split(/\s+/);
        for(my $j=0;$j<@a;$j++){
            $hash{$ind[$i]}{"k$j"}=$a[$j];
        }
        #my $line=join "\t",@a;
        #print O1 $ind[$i],"\t","$line\n";
        $i++;
    }
    foreach my $pop(@pop) {
        foreach my $indkey (sort keys %{$hash1{$pop}}) {
            foreach my $k (sort keys %{$hash{$indkey}}) {
                print O2 "$indkey\t$hash{$indkey}{$k}\t$k\n";
                print O1 $indkey,"\t","$hash{$indkey}{$k}\n";
            }
        }
    }
    close(F);
    close(O2);
    close(O1);
    print R "
a=read.table(\"$out2\",header=T)
a<-transform(a,id=factor(id,levels=unique(id)))
require(grid)
#m.a<-melt(a)
pdf(file=\"$prefix.$suffix.pdf\",width=80,height=14)
ggplot(a,aes(x=id,y=percent))+geom_bar(stat=\"identity\",aes(fill=k),width=1)+the
me(panel.background=element_blank(),axis.text.x=element_text(angle=270),axis.tick
s = element_blank(),legend.position =
\"none\",axis.title.x=element_blank(),axis.title.y=element_blank())
dev.off()
";
    print "$file complete\n";
}
```

```
close(R);

sub readPop{
    my $file=shift;
    chomp $file;
    my @pop;
    open(F,$file) || die "$!\n";
    while(<F>){
    chomp;
    if(/^(\S+)\s+(\S+)/){
      $hash1{$2}{$1}=$2;
      $pop{$2}++;
    }
    }
    close(F);
    @pop=sort keys %pop;
    #@pop=qw(LJ YMD WZ EMD WMD QQ ZG);
    #@pop=qw(CX LZ LJ YMD WZ EMD WMD QQ ZG);
    return @pop;
}
sub readPed{
    my $file=shift;
    chomp $file;
    my @r;
    open(F,$file) || die "$!\n";
    while(<F>){
    chomp;
    if(/^(\S+)/){
      push @r,$1;
    }
    }
    close(F);
    return @r;
}
```

最后的树型：


Europe
BS-1-ZL
BS2-1-ZL
HL03-1
MT-1-ZL
QL-1-ZL
QL2-1-ZL
SD-1-ZL
SN-1-ZL
SN2-1-ZL
TZ-1-ZL
XM-1-ZL
100
86
74
74
74
46
46
46
46
46

流程开发日志——粗枝云山（Picea asperata）

introduction

本文是我自己独立承担的第一个科研项目的科研记录，是关于青海不同地区的粗枝云杉物种的群体结构分析（目前正在做）以及更下游的生态学分析工作（后续会做，在计划之内），可以参考如下文章，类似工作的文章有：

https://onlinelibrary.wiley.com/doi/10.1111/mec.16758

数据的获取

数据是当时测得，现在都上传到了NCBI，需要用到一些操作把他下载下来：

```
conda install sra-tools

awk '{print "/user/WH/software/sra-tools/bin/prefetch "$1}' SRR_Acc_List.txt > run.sh
awk '{print "/user/WH/software/sra-tools/bin/fastq-dump --split-3 --gzip"$1"/"$1".sra"}' ../SRR_Acc_List.txt
```

如果报错：

```
fastq-dump --split-3 file.sra

An error occurred during processing.
A report was generated into the file '/root/ncbi_error_report.xml'.
If the problem persists, you may consider sending the file
to 'sra@ncbi.nlm.nih.gov' for assistance.
```

这是因为磁盘不够了，要么删一点你之前做的文件，要么加上gzip参数，一般都要压缩，以后下载的时候要养成习惯

最后下载的数据是63个样本，每个样本的测序数据都不大，大概都是1G的样子，采样图可以从Molecular Ecology之前师姐发的文章里面找到

过滤

服务器最近用的人有点多，想了个办法，分几批来并行算：

第一批 1789-1985

第二批 1986-2062

第三批 2065-2121

写两个shell脚本，一个脚本写具体的命令，一个脚本负责控制跑的批次

```
# 控制脚本
bash ./01.fastp_1.sh &
bash ./01.fastp_2.sh &
bash ./01.fastp_3.sh &  #这个文件建议在linux下生成，否则会有\r报错，也就是windows和linux的制表符不兼容
```

然后每个脚本里面大概是这样的：

```
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571789_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571789_2.fastq.gz -o
./SRR19571789_1_filt.fastq.gz -O ./SRR19571789_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571800_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571800_2.fastq.gz -o
./SRR19571800_1_filt.fastq.gz -O ./SRR19571800_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571811_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571811_2.fastq.gz -o
./SRR19571811_1_filt.fastq.gz -O ./SRR19571811_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571879_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571879_2.fastq.gz -o
./SRR19571879_1_filt.fastq.gz -O ./SRR19571879_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571881_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571881_2.fastq.gz -o
./SRR19571881_1_filt.fastq.gz -O ./SRR19571881_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571882_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571882_2.fastq.gz -o
./SRR19571882_1_filt.fastq.gz -O ./SRR19571882_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571883_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571883_2.fastq.gz -o
./SRR19571883_1_filt.fastq.gz -O ./SRR19571883_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571884_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571884_2.fastq.gz -o
./SRR19571884_1_filt.fastq.gz -O ./SRR19571884_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571900_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571900_2.fastq.gz -o
./SRR19571900_1_filt.fastq.gz -O ./SRR19571900_2_filt.fastq.gz &&
fastp -f 10 -F 10 -x -g -c -q 15 -u 40 -n 5 -i
/user/WH/Picea_asperata/data/SRR19571903_1.fastq.gz -I
/user/WH/Picea_asperata/data/SRR19571903_2.fastq.gz -o
./SRR19571903_1_filt.fastq.gz -O ./SRR19571903_2_filt.fastq.gz &&
```

每个fastp过滤用&&连起来，这样就保证每个shell分支只会提交一个任务，每次后台也只会有3个过滤任务运行，这样可以自动控制提交任务的数量

比对

这里选择用更多的处理器，而不是提交多个任务，原因是工作流比较长，与其分批次自动提交不如在比对的时候就多用几个线程把hisat的步骤给跑好，每个任务流如下：

```
hisat2 -p 40 -x /user/WH/workspace/Picea_asperata_0401/01.hisat2/reference/genome
-1 /user/WH/Picea_asperata/clean_data/SRR19571789_1_filt.fastq.gz -2
/user/WH/Picea_asperata/clean_data/SRR19571789_2_filt.fastq.gz -S
./sam/SRR19571789.sam &&
samtools view -@ 30 -F 4 -bS ./sam/SRR19571789.sam -o ./sam2bam/SRR19571789.bam
&&
samtools sort -@ 30 ./sam2bam/SRR19571789.bam -o ./sort/SRR19571789.sorted.bam &&
samtools index ./sort/SRR19571789.sorted.bam &&
```

标记pcr重复

使用gatk4.4.0.0软件进行标记,并将其去除，用picard也是一样的结果，因为picard包含在gatk4里面，除了命令改成了gatk之外其他没有任何区别，做这一步是参考了之前的论文，如果直接用gatk4的话是没有这一步骤的：

每次跑三个任务，还是用之前的办法控制任务数量：

```
bash 03.mkdup_1.sh &
bash 03.mkdup_2.sh &
bash 03.mkdup_3.sh & #这个文件建议在linux下生成，否则会有\r报错，也就是windows和linux的制
表符不兼容
```

一个run.sh如上，其他的参数设置如下：

```
gatk MarkDuplicates -I
/user/WH/workspace/Picea_asperata_0401/01.hisat2/sort/SRR19571986.sorted.bam -O
./SRR19571986_Markdup.bam -M metrics.txt --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP
1000 --REMOVE_DUPLICATES true &&
gatk MarkDuplicates -I
/user/WH/workspace/Picea_asperata_0401/01.hisat2/sort/SRR19571994.sorted.bam -O
./SRR19571994_Markdup.bam -M metrics.txt --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP
1000 --REMOVE_DUPLICATES true &&
gatk MarkDuplicates -I
/user/WH/workspace/Picea_asperata_0401/01.hisat2/sort/SRR19571997.sorted.bam -O
./SRR19571997_Markdup.bam -M metrics.txt --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP
1000 --REMOVE_DUPLICATES true &&
gatk MarkDuplicates -I
/user/WH/workspace/Picea_asperata_0401/01.hisat2/sort/SRR19571998.sorted.bam -O
./SRR19571998_Markdup.bam -M metrics.txt --MAX_FILE_HANDLES_FOR_READ_ENDS_MAP
1000 --REMOVE_DUPLICATES true &&
 ...
```

添加readgroup

使用picard，这里直接用的picard软件，是顺便下载picard验证了一遍是不是有Markduplicate这个工具

```
picard AddOrReplaceReadGroups --INPUT
/user/WH/workspace/Picea_asperata_0401/02.mkdup/SRR19572065_Markdup.bam --OUTPUT
./SRR19572065_ADD.bam --RGLB tLB_SRR19572065 --RGPL ILLUMINA --RGPU unit1 --RGSM
SRR19572065  &&
samtools index ./SRR19572065_ADD.bam
```

indel附近的重新比对

这里用的gatk3.8,因为gatk4没有这个功能，而且已发表的论文里面用的是这个，我这里全部都用的这个流程

做的时候发现之前的参考基因组的索引文件被删除了，所以这里重新构建索引:

```
gatk CreateSequenceDictionary -R my.ref.fa -O my.ref.dict
samtools faidx my.ref.fa
```

call SNP

使用samtools进行call SNP，流程和之前一样，然后过滤:

对于每个sample的VCF，过滤深度和质量值:

```perl
#!/usr/bin/env perl
use strict;
use warnings;
# 从命令行参数获取 VCF 文件路径和输出文件路径
my $vcf = shift;
my $out = shift;
# 设置过滤参数
my $min_dp = 10;              # 每个样本的最小读取深度，取的师哥的文章的值
my $max_dp = 500000;          # 每个样本的最大读取深度，这个取一个较大的就行，达不到的
my $min_qual = 30;            # 每个位点的最小质量分数，取的师哥的文章的值
# 使用 gzip 压缩输出文件
open O, "| gzip - > $out" or die "Cannot open gzip pipe to $out: $!";
# 使用 zcat 解压 VCF 文件
open I, "zcat $vcf |" or die "Cannot open pipe from $vcf: $!";
# 初始化变量
my $num;
my $min_dp_every_site = 0;  # 每个位点的最小读取深度
my $max_dp_every_site = 0;  # 每个位点的最大读取深度
# 逐行读取 VCF 文件
while (<I>) {
    chomp;
    # 如果是元数据行，直接打印到输出文件
    if (/^##/) {
        print O "$_\n";
    }
    # 如果是列标题行
    elsif (/^#/) {
        print O "$_\n";
        # 分割列标题行以获取样本数量
        my @head = split(/\s+/);
        $num = @head - 9;  # 样本数量
        # 根据样本数量计算每个位点的最小和最大读取深度
```

```
        $min_dp_every_site = $num * $min_dp;
        $max_dp_every_site = $num * $max_dp;
    }
    # 如果是数据行
    else {
        my @a = split(/\s+/);
        my ($chr, $pos, $id, $ref, $alt, $qual) = @a;  # 提取基本位点信息
        # 跳过替代序列长度不为 1 的位点
        next unless (length($alt) == 1);
        # 从第 8 列提取读取深度
        $a[7] =~ /DP=(\d+)/;
        my $dp = $1;
        # 跳过读取深度不在指定范围内的位点
        next unless ($dp >= $min_dp_every_site && $dp <= $max_dp_every_site);
        # 跳过质量分数不满足最小要求的位点
        next unless ($qual >= $min_qual);
        # 打印满足条件的位点到输出文件
        print O "$_\n";
    }
}
# 关闭文件句柄
close I;
close O;
```

过滤indel附近5bp的SNP：

```
#! /usr/bin/env perl
use strict;
use warnings;
my $vcf=shift;
my $out=shift;
open I,"zcat $vcf |";
my %indel;
while(my $line=<I>){
    next if($line=~/^#/);
    my $indel_test=0;
    my @a=split(/\s+/,$line);
    my ($chr,$pos,$id,$ref,$alt)=@a;
    if($line=~/INDEL/){
         $indel_test=1;
    }
    my $len=length($ref)-1;
    if($indel_test == 1){
        for(my $i=$pos-5;$i<=$pos+$len+5;$i++){
            $indel{$chr}{$i}=1;
        }
    }
}
close I;

open O,"| gzip > $out";
open I,"zcat $vcf |";
while (<I>) {
    chomp;
    if(/^#/){
```

```
        print O "$_\n";
    }
    else{
        my @a=split(/\s+/);
        my ($chr,$pos,$id,$ref,$alt)=@a;
        next if(exists $indel{$chr}{$pos});
        print O "$_\n";
    }
}
close I;
close O;
```

使用awk生成命令：

```
ls -l /user/WH/workspace/Picea_asperata_0401/call_SNP/05.call_snp/raw_vcf > raw_vcf.txt

awk '{print "perl filter_dp_qual.pl /user/WH/workspace/Picea_asperata_0401/call_SNP/05.call_snp/raw_vcf" $9}'

awk '{print "perl filter_indel.pl ./filt1/" $9 " ./filt2/" $9 }' ./filt1/filt1_vcf.txt > filter_indel.sh
```

这里用管道会无法输出

生成bcftools命令：

```
echo "bcftools merge" $(ls -1 | awk '{ORS=" "; print $9}')
```

在此之前，要压缩和生成索引：

```
awk '{print "bgzip " $9}' list.txt > bgzip.sh
awk '{print "tabix -p vcf " $9}' list.txt > tabix.sh
```

之后就可以合并了：

```
nohup ./bcftools_merge.sh > bcftools.log 2>&1 &
```

之后再过滤,过滤标准是老师给的：

```
vcftools --gzvcf ./merge.vcf.gz  --minDP 10 --max-missing-count 5 --recode --recode-INFO-all --stdout | gzip -c > merge.dp10ms5.vcf.gz &&
vcftools --gzvcf ./merge.dp10ms5.vcf.gz --min-alleles 2 --max-alleles 2 --non-ref-af 0.01 --recode --recode-INFO-all --stdout | gzip -c > merge.dp10ms5.snp.alte-2.vcf.gz
```

最终得到了merge.dp10ms5.snp.alte-2.vcf.gz，可以用这个vcf文件进行下游的群体结构分析

实验记录 群体分析新思路

现在暂时没有可以直接出来进化树和柱状图的方法，都是将两个图分别做出来然后拼在一起，因而两者能够对齐是一个工作量很大的工作，以往我是一点一点调节pop.txt的顺序，从而得到结果，现在发现一种新的办法，因而记录一下：

```
  ploto <-"myfile.o" # 树文件
  group_over <- "myfile.9.Q.group_over.txt" #样本分群文件
  group_col <- "myfile.9.Q.pca_color.txt" # 颜色匹配文件
  label_list <- "sample.list" #样本列表文件
  prefix <- ploto
```

```
# 包加载一波
library(ggtree)
library(ggplot2)
library(treeio)
library(tidytree)
library(pophelper)
library(patchwork)
```

```
# 上匹配颜色
lines <- readLines(group_col)
cells <- strsplit(lines, "\t")
data <- as.data.frame(do.call(rbind, cells), stringsAsFactors = FALSE)
mycolor <- data[-1,]
color_mapping <- with(mycolor, setNames(V2,V1))
```

ggtree进化树

```
tree<-read.newick(ploto)

# 两列一列 label 一列 group
group_info <- read.table(group_over,header = T)
colnames(group_info)[1] <- "label"
tree1<-full_join(tree,group_info,by="label")


ggtree_plot <- ggtree(tree1,aes(color=group),branch.length = "none")+
  layout_dendrogram()+
  theme(legend.position = "none")+
  scale_color_manual(values = color_mapping)
pdf(file = paste0(prefix,".tree.pdf"), width = 15, height = 3)
ggtree_plot
dev.off()
```

这段代码使用`ggtree`包绘制树形图。`aes(color = group)`设置了一个美学映射，即用`group`列的值（分组信息）来决定树的分支颜色。`branch.length = "none"`表示不显示分支长度。`layout_dendrogram()`设置了树形图的布局为树状图样式。`theme(legend.position = "none")`移除了图例。`scale_color_manual(values = color_mapping)`设置了一个手动的颜色映射，其中`color_mapping`是前面代码定义的，根据分组名称到颜色值的映射。

```
# 获取ggtree绘制的树的节点数据
data <- ggtree_plot$data

# 获取叶节点的顺序（按照y值排序，因为ggtree使用y值来决定节点在图上的位置）
itoltree_sort <- as.data.frame(data[data$isTip, ]$label[order(data[data$isTip,
]$y)])
colnames(itoltree_sort) <- "V1"
```

柱状图

```
# 画多个按照相同顺序
mul_sfiles<- list.files(path=getwd(),pattern = '*.Q$')
mul_sfiles <- mul_sfiles[order(nchar(mul_sfiles),mul_sfiles)]
mul_slist <- readQ(files = mul_sfiles)

label <- read.delim(label_list,header=F,stringsAsFactors=F,sep = '\t')
if(length(unique(sapply(mul_slist,nrow)))==1)
  mul_slist<-lapply(mul_slist,"rownames<-",label$V1)


sort_by_itoltree <- function(df, itoltree_sort) {
  return(df[match(itoltree_sort$V1, rownames(df)), ])
}

mul_slist <- lapply(mul_slist, function(df) {
  return(sort_by_itoltree(df, itoltree_sort))
})


plotQ(alignK(mul_slist[c(1,2)]),
      imgoutput="join",
      clustercol=clist$cb_set3,
      splabsize=5,
      basesize = 5,
      #splab=paste0("K=",sapply(mul_slist[length(mul_slist)],ncol)),
      splab=paste0("K=",sapply(mul_slist[c(1,2:length(mul_slist))],ncol)),
      sppos="left",
      showyaxis=F, #y轴
      showticks=F,
      panelspacer=0.1,
      exportplot=F,
      returnplot=T,
      #exportpath=getwd(),
      #sortind="all",
      showindlab=F,
      sharedindlab=F,
      indlabvjust =F,
      indlabsize = 1,
      showlegend = F,
      showtitle=F,
      #titlelab=key,
      #subtitlelab="population ",
      ordergrp=F, #颜色图例
      grplabalpha = 0,
```

```
      linealpha = 0,
      pointalpha = 0,
      #outputfilename=key,imgtype="pdf",
      #width = 15,
      #grplab=poly
)
```

拼接

```
library(ggplot2)
library(patchwork)

# 创建两个ggplot对象
p1 <- ggplot(...) + ...
p2 <- ggplot(...) + ...

# 使用patchwork拼接图形
p1 / p2
```

GO富集气泡图的绘制

```
library(plyr)
library(stringr)
library(grid)
library(ggplot2)
library(readxl)
df <- read_excel("C:/Users/ASUS/Desktop/rich_bubble/p1-p3.xls")
df$log_Pvalue <- -log10(df$PValue)
total_count <- sum(df$Count)
df$generatio <- df$Count / total_count
ggplot(df,aes(x = generatio, y=Term, size = Count, color = PValue)) +
  geom_point() +
  scale_size_area(max_size = 10) +
  scale_color_gradient(low = 'lightblue', high = 'darkblue') +
  theme_minimal() +
  labs(title = "Enrichment Bubble Chart", x = "Generation", y = "Term")
```

上面代码绘制的不对，下面的是对的

```
library(ggplot2)
library(dplyr)
library(tidyverse)
eGO <- read.table("C:/Users/ASUS/Desktop/rich_bubble/p2-p3.txt", header = TRUE,
sep = "\t") #文件编码要是utf8
total_count <- sum(eGO$Count)
eGO$generatio <- eGO$Count / total_count
eGOBP <- eGO %>%
  filter(ONTOLOGY=="BP") %>%
  arrange(desc(generatio)) %>%
  filter(row_number() >= 1,row_number() <= 20)
eGOCC <- eGO %>%
  filter(ONTOLOGY=="CC") %>%
  arrange(desc(generatio)) %>%
  filter(row_number() >= 1,row_number() <= 20)
eGOMF <- eGO %>%
  filter(ONTOLOGY=="MF") %>%
  arrange(desc(generatio)) %>%
  filter(row_number() >= 1,row_number() <= 20)
eGO10 <- rbind(eGOBP,eGOMF,eGOCC)
pdf(file="p2-p3.pdf",width = 10,height = 18)
ggplot(eGO10,aes(generatio,fct_reorder(factor(Term),generatio))) +
  geom_point(aes(size=Count,color=-1*log10(PValue))) +
  scale_color_gradient(low="green",high ="red") +
  labs(color=expression(-log[10](PValue)),size="Count", shape="ONTOLOGY",
       x="Gene Ratio",y="GO term",title="GO enrichment") +
  theme_bw() +
  facet_wrap( ~ ONTOLOGY,ncol= 1,scale='free')
dev.off()
```

可以肯定与适应性相关的：

leaf senescence

leaf morphogenesis

embryo development ending in seed dormancy

circadian regulation of gene expression

p1-p3

BP

上述列表中包含了许多生物学过程和调控途径，它们与生物体的适应性密切相关。生物体的适应性是指生物体对环境变化的响应和调整能力，以确保生存和繁衍。以下是一些与适应性特别相关的生物学过程：

1. **细胞对冷的响应 (cellular response to cold)**：这个过程中，生物体会调整其代谢途径和蛋白质表达，以适应低温环境。
2. **细胞对硒离子的响应 (cellular response to selenium ion)**：硒是一种重要的微量元素，对生物体的抗氧化防御系统至关重要。
3. **叶绿素生物合成过程 (chlorophyll biosynthetic process)**：叶绿素是植物进行光合作用的关键色素，其合成对于植物适应光照变化至关重要。
4. **光周期调控基因表达 (circadian regulation of gene expression)**：生物体通过昼夜节律来调整其生理活动，以适应日夜变化。
5. **防御反应到细菌 (defense response to bacterium)**：这是生物体对细菌感染的免疫反应，是适应性的重要组成部分。
6. **铁硫簇组装 (iron-sulfur cluster assembly)**：铁硫簇是许多酶的辅因子，对生物体的代谢和能量转换至关重要。
7. **叶形态发生 (leaf morphogenesis)**：叶片的形状和结构对植物适应环境变化（如光照、温度和湿度）非常重要。
8. **叶片衰老 (leaf senescence)**：叶片衰老是植物生命周期的一部分，与资源分配和再利用有关。
9. **响应到乙烯 (response to ethylene)**：乙烯是一种植物激素，对植物的生长、发育和对环境压力的响应有重要作用。
10. **响应到低温 (response to hypoxia)**：低氧响应是生物体对氧气供应不足的适应性反应。
11. **响应到金属离子 (response to metal ion)**：金属离子是许多生物过程的关键因素，生物体需要调节金属离子的摄取和利用。
12. **响应到渗透压应激 (response to osmotic stress)**：生物体需要对水分和盐分的变化做出响应，以维持细胞内环境的稳定。
13. **响应到温度刺激 (response to temperature stimulus)**：生物体对温度变化的适应性反应，对生存至关重要。
14. **响应到水分剥夺 (response to water deprivation)**：水是生命的基础，生物体必须适应水分不足的环境。
15. **淀粉代谢过程 (starch metabolic process)**：淀粉是植物储存能量的主要形式，其代谢对植物适应环境变化至关重要。
16. **根发育 (root development)**：根系的形态和功能对植物吸收水分和养分至关重要。
17. **种子萌发 (seedling development)**：种子萌发是植物生命周期的开始，对环境的适应性反应是其成功萌发的关键。

CC

在上述列表中，许多项指的是细胞内的结构或细胞器，这些结构或细胞器在生物体的适应性中扮演着重要角色。它们参与了多种生物学过程，包括能量转换、物质运输、蛋白质合成与降解、信号传导等，这些过程对于生物体适应环境变化至关重要。以下是一些与生物体适应性特别相关的细胞结构或细胞器：

1. **自噬体 (autophagosome)**：在营养匮乏或其他应激条件下，自噬体参与细胞内物质的回收利用，有助于细胞适应和生存。
2. **叶绿体 (chloroplast)**：叶绿体是植物细胞中进行光合作用的主要场所，对植物适应光照变化至关重要。
3. **细胞质 (cytoplasm)**：细胞质是细胞内进行许多代谢反应的场所，对细胞适应环境变化至关重要。
4. **细胞骨架 (cytoskeleton)**：细胞骨架参与维持细胞形态、细胞运动和物质运输，对细胞适应性有重要作用。
5. **内质网 (endoplasmic reticulum)**：内质网参与蛋白质的合成、折叠和运输，对细胞适应性至关重要。
6. **高尔基体 (Golgi apparatus)**：高尔基体参与蛋白质的后期修饰和运输，对细胞适应性有重要作用。
7. **线粒体 (mitochondria)**：线粒体是细胞的能量工厂，参与能量的产生和调节，对细胞适应性至关重要。
8. **核孔 (nuclear pore)**：核孔是核质交换的关键通道，对细胞适应性有重要作用。
9. **过氧化物酶体 (peroxisome)**：过氧化物酶体参与脂肪酸的β-氧化和过氧化物的分解，对细胞适应氧化应激有重要作用。
10. **植物型液泡 (plant-type vacuole)**：植物型液泡在植物细胞中负责储存物质、调节细胞内环境和响应环境变化。
11. **质膜 (plasma membrane)**：质膜是细胞与外界环境的界面，参与物质的进出和信号的传递，对细胞适应性至关重要。
12. **胞间连丝 (plasmodesma)**：胞间连丝在植物细胞间形成通道，有助于细胞间的通讯和物质交换，对植物适应性有重要作用。
13. **蛋白质酶体复合体 (proteasome complex)**：蛋白质酶体参与蛋白质的降解，对维持蛋白质稳态和细胞适应性至关重要。
14. **核糖体 (ribosome)**：核糖体是蛋白质合成的场所，对细胞适应性至关重要。
15. **溶酶体 (vacuole)**：溶酶体参与细胞内废物的降解和回收，对细胞适应性和维持细胞内环境稳定有重要作用。

这些细胞结构或细胞器通过参与不同的生物学过程，共同协作以使生物体能够适应环境变化。

MF

在上述列表中，许多活动与生物体的适应性密切相关，因为它们涉及到细胞内的关键生化过程和信号传导途径，这些过程和途径对于生物体在不断变化的环境中生存和繁衍至关重要。以下是20个与生物体适应性特别相关的活动：

1. **铁硫簇结合 (4 iron, 4 sulfur cluster binding)**：铁硫簇是许多酶的辅因子，对于能量代谢和应激反应至关重要。
2. **ATP结合 (ATP binding)**：ATP是细胞内能量转换的核心分子，其结合活性与细胞能量代谢和信号传导紧密相关。
3. **ATP酶活性 (ATPase activity)**：ATP水解提供细胞过程所需的能量，对于细胞适应性至关重要。

4. **ATPase活性，与物质跨膜运动耦合 (ATPase activity, coupled to transmembrane movement of substances)**：涉及细胞内物质的运输，对细胞内环境的维持和适应性有重要作用。
5. **受损DNA结合 (damaged DNA binding)**：DNA修复机制的一部分，对于细胞适应DNA损伤和防止突变非常重要。
6. **酶激活活性 (enzyme activator activity)**：调节酶活性，对代谢途径和信号传导途径的调控至关重要。
7. **GDP-岩藻糖跨膜转运蛋白活性 (GDP-fucose transmembrane transporter activity)**：涉及糖类的转运，对细胞表面分子的糖基化修饰有重要作用。
8. **谷胱甘肽γ-谷氨酰基-半胱氨酸转移酶活性 (glutathione gamma-glutamylcysteinyltransferase activity)**：参与谷胱甘肽的合成，对抗氧化应激和细胞保护有重要作用。
9. **GTP结合 (GTP binding)**：GTP是信号传导中的关键分子，参与蛋白质合成和细胞骨架的动态调控。
10. **GTP酶活性 (GTPase activity)**：GTP水解参与蛋白质的活性调控和信号传导。
11. **解旋酶活性 (helicase activity)**：DNA或RNA解旋是DNA复制和RNA转录的前提，对遗传信息的传递至关重要。
12. **金属离子结合 (metal ion binding)**：许多酶和蛋白质的功能依赖于金属离子，对细胞内多种生化过程至关重要。
13. **微管末端结合 (microtubule minus-end binding)**：涉及细胞骨架的动态调控，对细胞分裂和形态维持有重要作用。
14. **信使RNA结合 (mRNA binding)**：mRNA的稳定性和翻译效率对蛋白质合成和细胞适应性至关重要。
15. **蛋白酪氨酸/丝氨酸/苏氨酸磷酸酶活性 (protein tyrosine/serine/threonine phosphatase activity)**：磷酸酶在信号传导途径中去除磷酸基团，调节蛋白质活性。
16. **Rab GDP解离抑制因子活性 (Rab GDP-dissociation inhibitor activity)**：涉及G蛋白偶联的信号传导，对细胞内物质运输和膜泡运输有重要作用。
17. **RNA结合 (RNA binding)**：RNA的稳定性和功能对基因表达调控至关重要。
18. **RNA解旋酶活性 (RNA helicase activity)**：RNA解旋是RNA加工和翻译的前提，对RNA生物学至关重要。
19. **跨膜转运蛋白活性 (transmembrane transporter activity)**：物质跨膜运输对维持细胞内外环境平衡和适应性有重要作用。
20. **未折叠蛋白结合 (unfolded protein binding)**：涉及到蛋白质折叠和质量控制，对防止蛋白质错误折叠和聚集有重要作用。

这些活动涉及细胞内多个层面的生物化学过程，包括能量转换、物质运输、信号传导、DNA修复、蛋白质合成与降解等，它们共同构成了生物体适应性的分子基础。

p2-p3

BP

在上述列表中，许多生物学过程与生物体的适应性密切相关，因为它们涉及到细胞结构的维持、信号传导、能量代谢、生长发育、应激反应等关键生理功能。以下是20个与生物体适应性特别相关的生物学过程：

1. **肌动蛋白丝组织 (actin filament organization)**：肌动蛋白丝是细胞骨架的重要组成部分，对细胞形态和运动至关重要。
2. **细胞壁果胶生物合成过程 (cell wall pectin biosynthetic process)**：细胞壁的果胶成分对植物细胞的结构和防御机制有重要作用。
3. **细胞对电刺激的响应 (cellular response to electrical stimulus)**：在某些生物体中，如神经元，对电刺激的响应是其功能的关键部分。
4. **叶绿体分裂 (chloroplast fission)**：叶绿体的分裂和分布对植物细胞的能量转换和光合作用至关重要。
5. **昼夜节律调节基因表达 (circadian regulation of gene expression)**：昼夜节律的调节对生物体适应日常环境变化非常重要。
6. **细胞质分裂通过细胞板形成 (cytokinesis by cell plate formation)**：细胞分裂是生物体生长和繁殖的基础。
7. **防御反应到细菌 (defense response to bacterium)**：生物体的免疫反应是其适应性和生存的关键。
8. **DNA复制 (DNA replication)**：准确的DNA复制对维持遗传信息的完整性至关重要。
9. **双链断裂修复通过同源重组 (double-strand break repair via homologous recombination)**：DNA修复机制对维持基因组的稳定性和防止突变非常重要。
10. **胚胎发育以种子休眠结束 (embryo development ending in seed dormancy)**：种子休眠是植物适应不良环境的一种方式。
11. **内质网到高尔基体的囊泡介导运输 (ER to Golgi vesicle-mediated transport)**：囊泡运输是细胞内物质运输的关键途径。
12. **热适应 (heat acclimation)**：热适应是生物体对高温环境的响应，对其生存至关重要。
13. **铁硫簇组装 (iron-sulfur cluster assembly)**：铁硫簇在许多生物化学过程中扮演重要角色，如呼吸作用和光合作用。
14. **茉莉酸介导的信号传导途径 (jasmonic acid mediated signaling pathway)**：茉莉酸是一种植物激素，参与植物的防御反应和适应性。
15. **叶片衰老 (leaf senescence)**：叶片衰老是植物生命周期的一部分，与资源再分配和适应性有关。
16. **有丝分裂后期到液泡的运输 (late endosome to vacuole transport)**：囊泡运输在细胞内部物质的循环和降解中起关键作用。
17. **侧根形成 (lateral root formation)**：根系的发展对植物吸收水分和养分至关重要。
18. **线粒体分裂 (mitochondrial fission)**：线粒体的分裂和分布对细胞的能量代谢和适应性有重要作用。
19. **mRNA加工 (mRNA processing)**：mRNA的加工对基因表达的调控至关重要。
20. **蛋白质泛素化 (protein ubiquitination)**：蛋白质泛素化是蛋白质降解和调控的重要机制，对维持细胞内环境稳定和适应性有重要作用。

这些过程涵盖了从细胞结构的维持到信号传导、从能量代谢到生长发育、从DNA修复到应激反应等多个方面，它们共同构成了生物体适应性的复杂网络。

CC

在上述列表中，许多细胞结构和分子复合体对生物体的适应性至关重要，因为它们参与了细胞的基本功能和对环境变化的响应。以下是20个与生物体适应性特别相关的细胞结构和分子复合体：

1. **ATG1/ULK1激酶复合体 (ATG1/ULK1 kinase complex)**：在自噬过程中起关键作用，帮助细胞在营养不足时回收和再利用成分。
2. **自噬体 (autophagosome)**：自噬是细胞对营养剥夺或其他应激条件的适应性响应。
3. **细胞板 (cell plate)**：在植物细胞分裂过程中形成新的细胞壁，对细胞增殖至关重要。
4. **叶绿体 (chloroplast)**：进行光合作用，是植物适应光照变化和能量转换的关键细胞器。
5. **内质网 (endoplasmic reticulum)**：蛋白质合成和折叠的主要场所，对细胞适应性至关重要。
6. **高尔基体 (Golgi apparatus)**：参与蛋白质的后期修饰、排序和运输，对细胞的分泌活动和膜转运至关重要。
7. **细胞质 (cytoplasm)**：细胞内进行许多代谢反应的场所，对细胞适应性至关重要。
8. **细胞骨架 (cytoskeleton)**：维持细胞形态，参与细胞运动和物质运输，对细胞适应性有重要作用。
9. **内质网膜 (endoplasmic reticulum membrane)**：参与蛋白质的合成和脂质的生物合成，对细胞适应性至关重要。
10. **有丝分裂纺锤体 (mitotic spindle)**：在有丝分裂过程中分离染色体，对遗传信息的准确传递至关重要。
11. **核孔 (nuclear pore)**：调控物质进出细胞核，对细胞适应性有重要作用。
12. **过氧化物酶体 (peroxisome)**：参与脂肪酸的β-氧化和过氧化物的分解，对细胞适应氧化应激有重要作用。
13. **植物型液泡 (plant-type vacuole)**：在植物细胞中负责储存物质、调节细胞内环境和响应环境变化。
14. **质膜 (plasma membrane)**：细胞与外界环境的界面，参与物质的进出和信号的传递，对细胞适应性至关重要。
15. **胞间连丝 (plasmodesma)**：在植物细胞间形成通道，有助于细胞间的通讯和物质交换，对植物适应性有重要作用。
16. **核糖体 (ribosome)**：蛋白质合成的场所，对细胞适应性至关重要。
17. **信号识别粒子，叶绿体靶向 (signal recognition particle, chloroplast targeting)**：参与蛋白质的靶向和定位，对叶绿体功能至关重要。
18. **转录因子复合体 (transcription factor complex)**：调控基因表达，对细胞适应性有重要作用。
19. **泛素连接酶复合体 (ubiquitin ligase complex)**：参与蛋白质的泛素化和降解，对维持蛋白质稳态和细胞适应性至关重要。
20. **液泡膜 (vacuolar membrane)**：液泡在细胞内负责储存物质、调节细胞内环境和响应环境变化。

这些细胞结构和分子复合体通过参与不同的生物学过程，共同协作以使生物体能够适应环境变化。

MF

在上述列表中，许多酶活性和分子结合活动对于生物体的适应性至关重要，因为它们参与了细胞内的关键生化过程和信号传导途径。以下是20个与生物体适应性特别相关的酶活性和分子结合活动：

1. **4铁，4硫簇结合 (4 iron, 4 sulfur cluster binding)**：铁硫簇是许多电子传递蛋白的关键辅因子，对能量代谢和应激反应至关重要。
2. **ATP结合 (ATP binding)**：ATP是细胞内能量转换的核心分子，其结合活性与细胞能量代谢和信号传导紧密相关。
3. **ATPase活性 (ATPase activity)**：ATP水解提供细胞过程所需的能量，对于细胞适应性至关重要。

4. **ATPase活性，与离子跨膜运动耦合，磷酸化机制 (ATPase activity, coupled to transmembrane movement of ions, phosphorylative mechanism)**：涉及细胞内物质的运输，对细胞内环境的维持和适应性有重要作用。
5. **热休克蛋白结合 (chaperone binding)**：热休克蛋白帮助蛋白质正确折叠，对蛋白质稳态和细胞适应应激条件非常重要。
6. **组蛋白脱乙酰酶活性 (histone deacetylase activity)**：通过调控组蛋白的乙酰化状态，影响染色质结构和基因表达，与细胞适应性相关。
7. **金属离子结合 (metal ion binding)**：许多酶和蛋白质的功能依赖于金属离子，对细胞内多种生化过程至关重要。
8. **mRNA结合 (mRNA binding)**：mRNA的稳定性和翻译效率对蛋白质合成和细胞适应性至关重要。
9. **蛋白丝氨酸/苏氨酸激酶活性 (protein serine/threonine kinase activity)**：这类激酶在信号传导途径中磷酸化蛋白质，调节其活性，对细胞适应性至关重要。
10. **RNA结合 (RNA binding)**：RNA的稳定性和功能对基因表达调控至关重要。
11. **RNA解旋酶活性 (RNA helicase activity)**：RNA解旋是RNA加工和翻译的前提，对RNA生物学至关重要。
12. **跨膜受体蛋白激酶活性 (transmembrane receptor protein kinase activity)**：这类受体激酶参与细胞外信号的感知和传导，对细胞适应环境变化至关重要。
13. **泛素蛋白连接酶活性 (ubiquitin protein ligase activity)**：泛素化是蛋白质降解和调控的重要机制，对维持蛋白质稳态和细胞适应性至关重要。
14. **锌离子结合 (zinc ion binding)**：锌是许多转录因子和酶的辅助因子，对细胞内多种生化过程至关重要。
15. **甘油醛-3-磷酸脱氢酶(NAD+) (非磷酸化)活性 (glyceraldehyde-3-phosphate dehydrogenase (NAD+) (non-phosphorylating) activity)**：在糖酵解过程中起关键作用，对能量产生至关重要。
16. **磷脂酰肌醇结合 (phosphatidylinositol binding)**：磷脂酰肌醇是细胞信号传导中的重要分子，参与细胞适应性。
17. **钾离子通道活性 (potassium channel activity)**：钾离子通道对维持细胞膜电位和神经信号传递至关重要。
18. **蛋白同型二聚体形成活性 (protein homodimerization activity)**：蛋白质二聚体化是一种常见的调节机制，对信号传导和酶活性有重要作用。
19. **单链DNA结合 (single-stranded DNA binding)**：在DNA复制、修复和重组过程中起重要作用。
20. **SUMO激活酶活性 (SUMO activating enzyme activity)**：SUMO化是一种蛋白质翻译后修饰，对蛋白质的稳定性、亚细胞定位和功能有重要影响。

这些酶活性和分子结合活动涉及细胞内多个层面的生物化学过程，包括能量转换、物质运输、信号传导、蛋白质合成与降解、基因表达调控等，它们共同构成了生物体适应性的分子基础。

TBtools-p1-p3.enrich.temp.GO.Enrichment.final

Molecular function

在生物体的适应性中，许多分子功能都扮演着重要角色，包括但不限于蛋白质的合成、调控、运输、修复以及能量的转换等。以下是与生物体适应性相关的20个分子功能：

1. **translation repressor activity** - 控制基因表达，帮助生物体适应环境变化。

2. **ribosome binding** - 蛋白质合成的关键步骤，对生物体的生长和适应至关重要。
3. **molecular carrier activity** - 运输分子在细胞内或细胞间，对营养分配和信号传递至关重要。
4. **mRNA binding** - 与mRNA的结合有助于调控其稳定性和翻译效率，影响蛋白质的生产。
5. **carbohydrate kinase activity** - 参与糖的代谢，为生物体提供能量，适应不同环境。
6. **nuclear import signal receptor activity** - 调控蛋白质进入细胞核，影响基因表达和细胞功能。
7. **ligase activity, forming carbon-nitrogen bonds** - 参与DNA修复和重组，对维持基因组稳定性和适应性变化至关重要。
8. **isomerase activity** - 催化异构体之间的转换，可能影响代谢途径和蛋白质结构。
9. **RNA helicase activity** - 解旋RNA，对RNA的加工和翻译至关重要。
10. **phosphatase activity** - 调控磷酸化状态，影响蛋白质的活性和细胞信号传递。
11. **GTPase regulator activity** - 调控GTP酶活性，涉及多种细胞过程，包括信号传导和蛋白质合成。
12. **snoRNA binding** - 小核仁RNA的结合可能参与rRNA的加工，影响核糖体的生物合成。
13. **ATP-dependent activity, acting on RNA** - ATP依赖的RNA加工，对RNA的稳定性和功能至关重要。
14. **phosphoric ester hydrolase activity** - 参与磷酸酯的水解，对能量转换和信号传递有重要作用。
15. **enzyme regulator activity** - 调控酶的活性，影响代谢途径和生物体的适应性。
16. **peptidase activity** - 蛋白质降解，对营养循环和信号传递至关重要。
17. **molecular adaptor activity** - 作为信号传递的中介，帮助生物体响应外界信号。
18. **phosphoprotein phosphatase activity** - 调控磷酸化蛋白质的去磷酸化，影响蛋白质功能和细胞信号。
19. **translation regulator activity** - 调控翻译过程，影响蛋白质的合成和生物体的适应性。
20. **nucleocytoplasmic carrier activity** - 调控分子在细胞核和细胞质之间的运输，对基因表达调控至关重要。

这些分子功能在生物体的适应性中扮演着不同的角色，从基因表达的调控到能量的转换和信号传递，都是生物体适应环境变化的关键因素。

Cellular component

在上述列表中，许多术语指的是细胞器、细胞结构或细胞过程，这些对于生物体的适应性至关重要。以下是20个与生物体适应性相关的项目：

1. **90S preribosome (90S核糖体前体)** - 在核糖体生物合成中起作用，对蛋白质合成至关重要。
2. **Amyloplast (淀粉体)** - 在植物细胞中参与淀粉的合成和储存，与能量储存相关。
3. **Autophagosome (自噬体)** - 参与自噬，帮助细胞回收和再利用组分，对适应营养变化至关重要。
4. **Cell Plate (细胞板)** - 在植物细胞分裂中形成新的细胞壁，对细胞增殖和组织发展至关重要。
5. **Chloroplast (叶绿体)** - 进行光合作用，是植物能量转换和适应光照条件的关键。
6. **Chloroplast Envelope (叶绿体包膜)** - 保护叶绿体内部结构，调节物质交换。
7. **Chloroplast Stroma (叶绿体基质)** - 叶绿体内部的液体部分，光合作用发生的地方。
8. **Chloroplast Thylakoid (叶绿体类囊体)** - 含有光合作用所需的色素和酶。
9. **Cis-Golgi Network (顺式高尔基网络)** - 高尔基体的一部分，涉及蛋白质的修饰和运输。
10. **Endoplasmic Reticulum (内质网)** - 蛋白质和脂质的合成场所，对细胞的代谢活动至关重要。
11. **Endosome (内体)** - 参与细胞内物质的运输和处理。

12. **Golgi Apparatus (高尔基体)** - 参与蛋白质的后期修饰、包装和分泌。
13. **Intracellular Vesicle (细胞内囊泡)** - 在细胞内运输物质和信号分子。
14. **Mitochondrial Matrix (线粒体基质)** - 线粒体内部的液体环境，是某些代谢途径的场所。
15. **Nucleoid (核体)** - 在某些原核生物中作为遗传物质的储存区域。
16. **Nucleus (细胞核)** - 包含遗传信息，是细胞的控制中心。
17. **Peroxisome (过氧化物酶体)** - 参与脂肪酸的氧化和有毒物质的解毒。
18. **Plastid (质体)** - 植物细胞中的一种双层膜结构，包括叶绿体和有色体。
19. **Protein Storage Vacuole (蛋白质存储液泡)** - 储存蛋白质和其他物质，对细胞的营养状态和适应性有重要作用。
20. **Thylakoid (类囊体)** - 在叶绿体和某些细菌中，参与光合作用或能量转换。

这些结构和过程对于细胞的正常功能和生物体对环境变化的适应性至关重要。

Biological process

在上述列表中，有许多生物学过程和分子功能与生物体的适应性密切相关。以下是20个我认为最能代表其对生物体适应性密切相关的过程：

1. **ribosome biogenesis (核糖体生物发生)** - 核糖体是蛋白质合成的场所，其生物合成对细胞功能至关重要。
2. **rRNA processing (rRNA加工)** - rRNA是核糖体的核心组成部分，其加工对核糖体的成熟和功能至关重要。
3. **cellular component biogenesis (细胞组分生物发生)** - 涉及细胞内各种结构和器官的合成，对维持细胞结构和功能至关重要。
4. **ribonucleoprotein complex biogenesis (核糖核蛋白复合体生物发生)** - 核糖核蛋白复合体在RNA加工和蛋白质合成中起关键作用。
5. **rRNA metabolic process (rRNA代谢过程)** - rRNA的代谢对于核糖体的合成和功能至关重要。
6. **cellular localization (细胞定位)** - 蛋白质和其他分子的正确定位对于细胞功能和信号传导至关重要。
7. **protein localization (蛋白质定位)** - 确保蛋白质在正确的细胞位置执行其功能。
8. **cellular macromolecule localization (细胞大分子定位)** - 细胞内大分子的正确定位对于细胞结构和功能至关重要。
9. **cellular component organization or biogenesis (细胞组分组织或生物发生)** - 涉及细胞内结构的组织和形成，对细胞适应性至关重要。
10. **ncRNA processing (非编码RNA加工)** - 非编码RNA在调控基因表达和蛋白质合成中起重要作用。
11. **establishment of protein localization (蛋白质定位的建立)** - 蛋白质定位的建立对于细胞内信号传导和功能至关重要。
12. **RNA processing (RNA加工)** - RNA加工对于蛋白质合成和基因表达调控至关重要。
13. **ncRNA metabolic process (非编码RNA代谢过程)** - 非编码RNA在调控细胞过程和适应性中起关键作用。
14. **Golgi organization (高尔基体组织)** - 高尔基体在蛋白质修饰、包装和运输中起关键作用。
15. **biosynthetic process (生物合成过程)** - 生物合成过程对于细胞内各种分子的合成至关重要。
16. **protein transport (蛋白质运输)** - 蛋白质运输对于蛋白质在细胞内的正确定位和功能至关重要。
17. **organelle organization (细胞器组织)** - 细胞器的正确组织对于细胞功能和适应性至关重要。

18. **intracellular transport (细胞内运输)** - 细胞内物质的运输对于细胞代谢和适应性至关重要。
19. **vesicle-mediated transport (囊泡介导的运输)** - 囊泡运输对于物质在细胞内的分配和细胞器间通信至关重要。
20. **regulation of gene expression (基因表达的调控)** - 基因表达的适当调控对于细胞对环境变化的适应性至关重要。

这些过程涵盖了从分子合成、加工、定位到细胞内运输和基因表达调控等多个层面，共同支持生物体对环境变化的适应性。

TBtools-p2-p3.enrich.temp.GO.Enrichment.final

Molecular function

生物体的适应性涉及到多种生物学过程和分子机制，以确保生物体在不同的环境条件下生存和繁衍。以下是与生物体适应性密切相关的20种酶和蛋白质活性：

1. **Hsp70 protein binding** - 热休克蛋白70的结合能力，对于应对热应激和蛋白质折叠至关重要。
2. **protein-folding chaperone binding** - 蛋白质折叠伴侣的结合，帮助蛋白质正确折叠，防止错误折叠和聚集。
3. **ATPase binding** - ATP酶的结合，参与能量转换和细胞内运输过程。
4. **metallopeptidase activity** - 金属肽酶活性，参与蛋白质的降解，对营养吸收和应激反应有重要作用。
5. **catalytic activity, acting on RNA** - 催化RNA的活性，可能涉及RNA的加工和降解，对基因表达调控至关重要。
6. **MF activator activity** - 调节因子(MF)的激活活性，可能涉及转录因子的激活，影响基因表达。
7. **histone deacetylase activity** - 组蛋白去乙酰化酶活性，调控染色质结构和基因表达。
8. **L-malate dehydrogenase activity** - L-苹果酸脱氢酶活性，参与能量代谢和应激反应。
9. **catalytic activity, acting on a nucleic acid** - 作用于核酸的催化活性，可能涉及DNA修复和复制。
10. **GTPase regulator activity** - GTP酶调节活性，影响细胞信号传导和蛋白质定位。
11. **hydrolase activity, acting on carbon-nitrogen (but not peptide) bonds, in cyclic amidines** - 作用于碳氮键（非肽键）的环状酰胺的酶活性，可能涉及某些生物合成和降解过程。
12. **ubiquitin-like protein binding** - 类泛素蛋白的结合，参与蛋白质的降解和调控。
13. **oxidoreductase activity, acting on the CH-NH group of donors, NAD or NADP as acceptor** - 作用于CH-NH基团供体的氧化还原酶活性，涉及能量代谢。
14. **translation factor activity, RNA binding** - RNA结合的翻译因子活性，调控蛋白质合成。
15. **endopeptidase activity** - 内肽酶活性，参与蛋白质的降解和信号传导。
16. **ribonucleoside triphosphate phosphatase activity** - 核糖核苷酸三磷酸磷酸酶活性，参与RNA合成和修复。
17. **enzyme activator activity** - 酶激活活性，可能涉及多种代谢途径的调控。
18. **isomerase activity** - 异构酶活性，参与代谢过程中的分子结构转换。
19. **heat shock protein binding** - 热休克蛋白结合，帮助蛋白质正确折叠，应对热应激。

20. **ATP-dependent activity, acting on DNA** - 作用于DNA的ATP依赖性活性，可能涉及DNA修复和复制。

这些活性在种子休眠和萌发过程中特别重要，因为它们涉及到能量代谢、蛋白质合成和降解、以及对环境变化的响应。通过这些机制，植物能够适应不同的环境条件，确保在适宜的时候萌发和生长。

Cellular component

在生物体的适应性中，特别是在种子休眠和萌发的过程中，许多细胞器和细胞结构扮演着关键角色。以下是与生物体适应性密切相关的20个细胞器和细胞结构：

1. **chloroplast** - 叶绿体，负责光合作用，对植物能量供应至关重要。
2. **cytoplasm** - 细胞质，细胞内进行许多生化反应的场所。
3. **endoplasmic reticulum** - 内质网，参与蛋白质和脂质的合成。
4. **Golgi apparatus** - 高尔基体，负责蛋白质的加工、修饰和分泌。
5. **mitochondrion** - 线粒体，细胞的能量工厂，负责ATP的产生。
6. **nucleus** - 细胞核，含有遗传信息，控制细胞的遗传活动。
7. **nucleolus** - 核仁，参与核糖体的合成。
8. **organelle envelope** - 细胞器膜，界定细胞器的结构和功能。
9. **peroxisome** - 过氧化物酶体，参与脂肪酸的氧化和分解。
10. **plastid** - 质体，植物细胞中的一种细胞器，除了叶绿体外的其他类型。
11. **protein-containing complex** - 蛋白质复合体，许多生物学过程的核心。
12. **vacuole** - 液泡，储存水分、营养物质和废物，调节细胞内环境。
13. **cytosol** - 细胞质基质，细胞质中的流体部分，许多代谢反应在此进行。
14. **nucleoplasm** - 核基质，细胞核内的基质，参与核内结构的维持。
15. **organelle subcompartment** - 细胞器亚区室，细胞器内部的特定功能区域。
16. **endoplasmic reticulum subcompartment** - 内质网亚区室，内质网的特定功能区域。
17. **Golgi subcompartment** - 高尔基体亚区室，高尔基体的特定功能区域。
18. **mitochondrial matrix** - 线粒体基质，线粒体内部的区域，进行某些代谢途径。
19. **chloroplast stroma** - 叶绿体基质，叶绿体内部的液体区域，光合作用的一部分在此进行。
20. **plastid stroma** - 质体基质，质体内部的区域，进行特定的生化反应。

这些细胞器和细胞结构在种子的休眠和萌发过程中发挥着关键作用，它们参与能量代谢、蛋白质合成与加工、脂质合成、废物处理、信号传导、遗传信息的表达与调控等多个方面，共同确保种子能够在适宜的条件下成功萌发并生长。

Biological process

在生物体的适应性中，特别是在种子休眠和萌发的过程中，细胞内物质的运输和定位对于细胞功能和生物体的存活至关重要。以下是与生物体适应性密切相关的20个细胞过程：

1. **cellular localization** - 细胞定位，确保蛋白质和细胞器在正确的位置发挥功能。
2. **vesicle-mediated transport** - 囊泡介导的运输，物质在细胞内及细胞器之间的运输。
3. **establishment of localization in cell** - 细胞内定位的建立，涉及细胞内结构的正确分布。
4. **protein localization** - 蛋白质定位，蛋白质在细胞内的正确位置对于其功能至关重要。
5. **establishment of protein localization** - 蛋白质定位的建立，确保蛋白质到达其功能位点。

6. **macromolecule localization** - 大分子定位，包括蛋白质、核酸等在细胞内的分布。
7. **cellular macromolecule localization** - 细胞内大分子定位，对细胞功能至关重要。
8. **protein transport** - 蛋白质运输，蛋白质在细胞内外的运输过程。
9. **establishment of localization** - 定位的建立，涉及细胞内结构和成分的正确分布。
10. **intracellular transport** - 细胞内运输，细胞器和分子在细胞内的运动。
11. **organelle organization** - 细胞器组织，细胞器的结构和功能维持。
12. **nitrogen compound transport** - 氮化合物运输，氮是生物体生长和蛋白质合成的关键元素。
13. **cellular biosynthetic process** - 细胞生物合成过程，合成细胞所需的分子。
14. **proteolysis** - 蛋白质降解，细胞通过降解不再需要的蛋白质来回收氨基酸。
15. **regulation of protein metabolic process** - 蛋白质代谢过程的调节，控制蛋白质的合成和降解。
16. **organic substance transport** - 有机物质运输，细胞内外物质交换的关键过程。
17. **transport** - 运输，细胞生命活动的基本过程之一。
18. **heat acclimation** - 热适应，生物体对热环境的适应性反应。
19. **organic substance biosynthetic process** - 有机物质生物合成过程，合成细胞所需的有机分子。
20. **response to heat** - 对热的响应，生物体对高温环境的适应性反应。

这些过程在种子休眠和萌发期间特别重要，因为它们涉及到细胞内物质的合成、运输、定位和降解，这些活动共同确保种子在适宜的条件下能够成功萌发并适应环境变化。

实验记录 构建基因树

使用treebest软件进行基因树（即针对某一个基因的每个个体的发育关系）的构建，命令如下：

```
nohup treebest nj -b 100 103342.vcf.fa > 103342.vcf.nj
```

eggNOG数据库本地化

COG功能分成四大类，信息存储和处理（information storage and processing）、细胞过程和信号（cellular processes and signaling）、代谢(metabolism) 和缺失的功能描述（poorly characterized）

而eggNOG数据库是COG数据库的增强版，有在线网站可以直接分析，而最近网站在维护，因而考虑自己搭建本地环境来进行分析。

在线分析网站：http://eggnogdb.embl.de/#/app/emapper

首先下载软件:

```
wget https://github.com/eggnogdb/eggnog-mapper/archive/1.0.3.tar.gz
tar -zxvf 1.0.3.tar.gz

#下载eggNGO数据库
wget http://eggnogdb.embl.de/download/emapperdb-5.0.0/eggnog.db.gz
wget http://eggnogdb.embl.de/download/emapperdb-5.0.0/eggnog_proteins.dmnd.gz

#下载软件
wget https://github.com/bbuchfink/diamond/releases/download/v2.1.9/diamond-
linux64.tar.gz
tar -zxvf diamond-linux64.tar.gz
wget http://eddylab.org/software/hmmer/hmmer-3.3.2.tar.gz
tar -zxvf hmmer-3.3.2.tar.gz
```

然后运行：

```
nohup python2 /user/WH/database/eggNGO/eggnog-mapper-1.0.3/emapper.py -i
my.ref.fa -o out -m diamond --cpu 32 --seed_ortholog_evalue 1e-5 --dmnd_db
/user/WH/database/eggNGO/eggnog_proteins.dmnd > out.log 2>&1 &
```

报错:

```
Annotation database data/eggnog.db not present. Use download_eggnog_database.py
to fetch it
```

考虑是安装的问题，换用conda试一下：

```
conda install eggnog-mapper
conda install -c bioconda diamond
conda install -c bioconda hmmer
```

此时conda已经自动配置好了环境变量，因而可以直接：

```
download_eggnog_data.py
```

之后可以进行分析了：

```
emapper.py -i my.ref.fa -o out -m diamond --cpu 32 --seed_ortholog_evalue 1e-5 --
dmnd_db eggnog_proteins.dmnd
```

最后得到的表格解读如下：

query_name：输入基因id

seed_eggNOG_ortholog：eggNOG中的最佳蛋白质匹配

seed_ortholog_evalue：evalue

seed_ortholog_score：score值

predicted_gene_name：预测的基因名称

GO_terms：GO功能信息

KEGG_pathways：KEGG功能信息

Annotation_tax_scope：注释的物种范围

OGs：eggNOG直系同源群列表

bestOG|evalue|score：最佳匹配直系组（仅在HMM模式下）

COG cat：从最佳匹配OG推断出的COG功能类别

eggNOG annot：功能描述

脚本编写 基因树vcf2fasta前置工作

在完成了基因的选择之后，想观察具体个体之间的流向情况，这里我们选择构建基因树看一下结果，目的就是得到每一个基因的对应个体的对应片段，在一个fasta文件中，从而构建基因树，因而这个问题的难点就是要将数据进行归类，具体的思路就是，从总的vcf文件中，找到所有该基因的序列，再提取每个个体的片段，分装在不同的fasta中，因为vcf中有基因的信息以及sample的信息，因而可以实现如下：

```
#从总vcf文件中提取某个contig的vcf文件

#这一步是从call出来的总vcf文件提出出对应基因名称的vcf，其为SNPs信息
1./user/Mahx/ruanjian/bcftools-1.10/bcftools view -r comp72876_c0_seq1
all_bre_dp10ms5.snp.alte-2.vcf.gz > 72876.vcf
#对提取出的vcf文件进行bgzip压缩和索引构建
2.bgzip 72876.vcf
3.tabix -p vcf 72876.vcf.gz
#1.txt内存储需要提取的基因名称，此步骤旨在从参考基因组中提取出对应的基因组成fasta文件
4./user/Mahx/baiqian/fdM/4-15-go/p1-p3/faSomeRecords /user/Mahx/baiqian/my.ref.fa
1.txt 76668.fasta
#将对应基因的参考基因组fasta和call出来的个体的vcf文件进行比对，从而得到含有对应个体变异信息的
fasta文件
5./user/Mahx/ruanjian/bcftools-1.10/bcftools consensus -f 72876.fasta
72876.vcf.gz --sample rc14-2  > rc14-2.fa
#最后：rc14-2.fa文件中的>comp72876_c0_seq1行表示这是comp72876_c0_seq1这个基因在rc14-2个
体中的序列。
#然后，将所有的个体都按以上步骤进行处理，每个基因都可以得到121个fasta文件
individuals="rc14-2 rc14-6 rc14-9 rc2-2 rc2-6 rc2-9 rc3-1 rc3-4 rc3-5 rc3-8 rc4-1
rc4-2 rc4-3 rc4-4 rc4-5 rc4-7 rc5-10 rc5-1 rc5-3 rc5-4 rc5-9 rc6-1 rc6-2 rc6-4
rc6-7 rc6-9 rc7-10 rc7-1 rc7-3 rc7-9 RL13-1 RL13-2 RL13-3 RL13-4 RL13-5 RL13-6
RL13-7 RL13-8 RL22-1 RL22-6 RL22-7 RL24-10 RL24-3 RL24-5 RL24-7 RL24-9 RL27-2
RL27-5 RL27-6 RL35-2 RL35-3 RL35-5 RL35-6 RL39-10 RL39-1 RL39-2 RL39-4 RL39-5
RL39-6 RL39-7 RL39-8 RL39-9 zlc04-10 zlc04-1 zlc04-2 zlc04-3 zlc04-4 zlc04-5
zlc04-6 zlc04-8 zlc04-9 zlc07-10 zlc07-1 zlc07-3 zlc07-4 zlc07-6 zlc07-7 zlc07-8
zlc07-9 zlc09-11 zlc09-13 zlc09-15 zlc09-16 zlc09-17 zlc09-18 zlc09-3 zlc09-5
zlc09-6 zlc09-7 zlc09-8 zlc09-9 zlc10-10 zlc10-1 zlc10-2 zlc10-3 zlc10-4 zlc10-5
zlc10-6 zlc10-7 zlc10-8 zlc10-9 zlc23-4 zlc23-6 zlc23-7 zlc29-1 zlc29-2 zlc29-4
zlc30-3 zlc30-5 zlc30-8 zlc35-1 zlc35-2 zlc35-3 zlc35-5 SRR9595769"
#使用sed命令将fasta标签替换成对应的个体名称，方便后面构建基因树
for individual in $individuals; do     sed -i
"s/>comp72876_c0_seq1/>${individual}_comp72876_c0_seq1/g" ${individual}.fa; done
#将fasta进行合并
cat *.fa > all-72876.fa
```

将上述的流程串起来，可以得到以下代码：

```
import os
#保证gene_tree_build.py,asp-intro-PSG-9.txt,all_bre_dp10ms5.snp.alte-2.vcf.gz在一个
目录下
current = '.'
adaptive_gene = []
```

```
sample = "rc14-2 rc14-6 rc14-9 rc2-2 rc2-6 rc2-9 rc3-1 rc3-4 rc3-5 rc3-8 rc4-1
rc4-2 rc4-3 rc4-4 rc4-5 rc4-7 rc5-10 rc5-1 rc5-3 rc5-4 rc5-9 rc6-1 rc6-2 rc6-4
rc6-7 rc6-9 rc7-10 rc7-1 rc7-3 rc7-9 RL13-1 RL13-2 RL13-3 RL13-4 RL13-5 RL13-6
RL13-7 RL13-8 RL22-1 RL22-6 RL22-7 RL24-10 RL24-3 RL24-5 RL24-7 RL24-9 RL27-2
RL27-5 RL27-6 RL35-2 RL35-3 RL35-5 RL35-6 RL39-10 RL39-1 RL39-2 RL39-4 RL39-5
RL39-6 RL39-7 RL39-8 RL39-9 zlc04-10 zlc04-1 zlc04-2 zlc04-3 zlc04-4 zlc04-5
zlc04-6 zlc04-8 zlc04-9 zlc07-10 zlc07-1 zlc07-3 zlc07-4 zlc07-6 zlc07-7 zlc07-8
zlc07-9 zlc09-11 zlc09-13 zlc09-15 zlc09-16 zlc09-17 zlc09-18 zlc09-3 zlc09-5
zlc09-6 zlc09-7 zlc09-8 zlc09-9 zlc10-10 zlc10-1 zlc10-2 zlc10-3 zlc10-4 zlc10-5
zlc10-6 zlc10-7 zlc10-8 zlc10-9 zlc23-4 zlc23-6 zlc23-7 zlc29-1 zlc29-2 zlc29-4
zlc30-3 zlc30-5 zlc30-8 zlc35-1 zlc35-2 zlc35-3 zlc35-5 SRR9595769".split(" ")
with open("mey_cra-intro-PSG-3.txt",'r') as file:
    for line in file:
        gene = line.strip()
        adaptive_gene.append(gene)

os.makedirs("./txt", exist_ok=True)
for i in range(len(adaptive_gene)):
    txt = os.path.join("./txt","{}.txt".format(adaptive_gene[i]))
    with open(txt,"w") as file:
        file.write(adaptive_gene[i])

output_file = os.path.join(current, "mey_cra-intro-PSG-3.sh")
with open(output_file, "w") as file:
    for i in range(len(adaptive_gene)):
        file.write("/user/Mahx/ruanjian/bcftools-1.10/bcftools view -r {0}
./all_bre_dp10ms5.snp.alte-2.vcf.gz > ./{0}/{0}.vcf
&&\n".format(adaptive_gene[i]))
        file.write("/user/Mahx/ruanjian/tabix-0.2.6/bgzip ./{0}/{0}.vcf
&&\n".format(adaptive_gene[i]))
        file.write("/user/Mahx/ruanjian/tabix-0.2.6/tabix -p vcf ./{0}/{0}.vcf.gz
&&\n".format(adaptive_gene[i]))
        file.write("/user/Mahx/baiqian/fdM/4-15-go/p1-p3/faSomeRecords
/user/Mahx/baiqian/my.ref.fa ./txt/{0}.txt ./{0}/{0}.fasta
&&\n".format(adaptive_gene[i]))
        for j in range(len(sample)):
            file.write("/user/Mahx/ruanjian/bcftools-1.10/bcftools consensus -f
./{0}/{0}.fasta ./{0}/{0}.vcf.gz --sample {1} > ./{0}/{1}-{0}.fa
&&\n".format(adaptive_gene[i],sample[j]))
            file.write("sed -i 's/>{1}/>{0}/g' ./{1}/{0}-{1}.fa
&&\n".format(sample[j],adaptive_gene[i]))
        file.write("cat ")
        for j in range(len(sample)):
            file.write("./{1}/{0}-{1}.fa ".format(sample[j],adaptive_gene[i]))
        file.write("> all-{0}.fa &&\n".format(adaptive_gene[i]))
```

运行得到对应的txt文件夹以及sh文件，在对应目录下运行即可。