转录组分析

生信帮
云南农业大学
云南省生物大数据重点实验室
2023年8月6日

目录

第一章 RNAseq测序原理介绍
第二章 数据质量控制
第三章 RPKM、FPKM、TPM的区别
第四章 定量分析（hisat2 + stringtie）
第五章 Ballgown差异分析

目录

第六章 DESeq2差异分析
第七章 edgeR差异分析
第八章 火山图绘制
第九章 venn图绘制
第十章 热图绘制
第十一章 WGCNA分析

RNAseq测序原理介绍

随着后基因组学的到来，转录组学、蛋白质组学、代谢组学等各种组学技术相继出现，其中转录组学是率先发展起来以及应用最广泛的技术。转录组，广义上指特定组织或细胞在某一发育阶段或功能状态下转录出来的所有RNA的总和，包含mRNA、rRNA、tRNA与其他非编码RNA；狭义上指所有mRNA的集合。

转录组测序（RNA-Seq）是针对基因组上起到转录作用的区间进行测序。其技术原理主要是从总RNA中通过ployA富集单链的mRNA，经反转录得到双链cDNA，然后对cDNA进行高通量测序过程。RNA-Seq具有操作简单、不局限于已知的基因组序列信息，不需要克隆、可获得低丰度表达基因、通量高、灵敏度高、成本低及应用领域广等优点。

转录组测序（RNA-Seq）对研究关键调控基因，发现新转录本，寻找不同处理条件下与差异性状或表型相关联的主要功能基因，寻找时序性变化、空间特异性变化的主控因素，在转录水平上进行性状定位、进化生物学研究等提供技术支持。

根据是否有参考基因组信息，可分为有参转录组和无参转录组测序。

RNAseq测序原理介绍

RNA-Seq是一种集合实验方法和计算机手段为一体的技术。RNA-Seq中的实验方法涉及从细胞或组织样本中分离RNA、构建不同RNA文库、对文库进行化学法测序以及随后的生物信息学数据分析。

RNA-Seq方法的产生源自于测序技术的世代革新。

第一代测序技术又称Sanger测序，指Sanger双脱氧链终止法测序，一次运行可产生10万条600-1000bp的碱基序列。

特点：1.准确；2.灵敏度较低；3.通量小，成本高；4.只能检测到部分突变形式。

RNAseq测序原理介绍

第二代测序技术也称下一代测序(NGS)或高通量测序技术。一次运行可产生60亿条100bp的碱基序列。

优势：1.通量高，成本低；2.灵敏度高；3.速度快；

缺点：读长短；

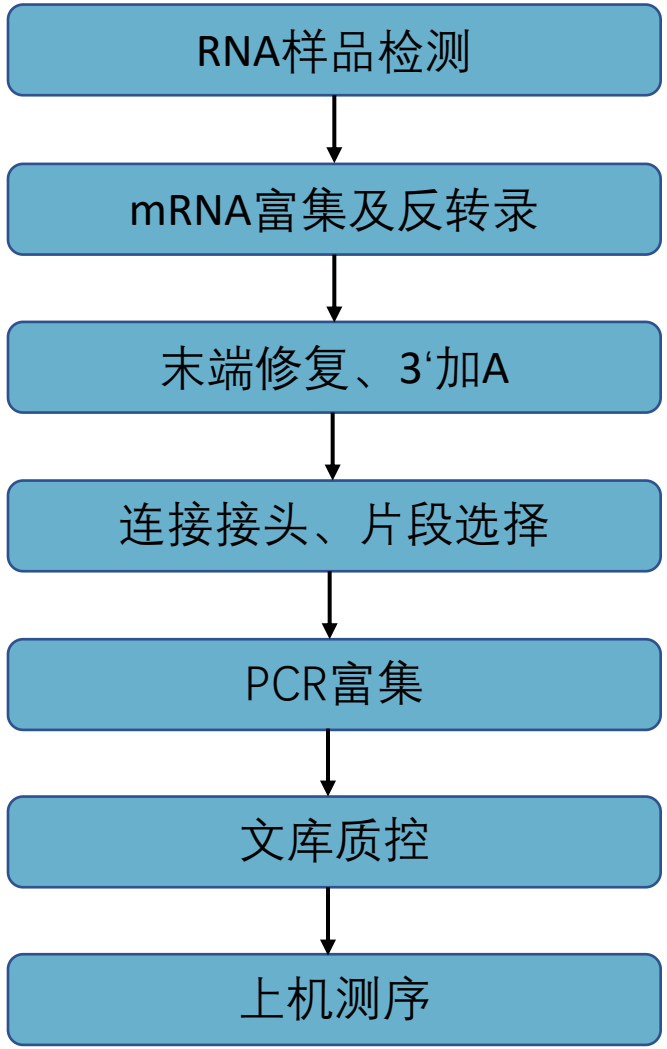
第三代测序技术是指单分子测序技术，也叫从头测序技术，同样是通过化学合成法进行大规模边合成边测序，不同之处在于，可以实现对每一条DNA或RNA分子进行单独测序。每个反应所读取的序列长度（reads）变得更长，甚至可达到10000个核苷酸。

优势： 1.通量高；2.灵敏度高；3.速度快；4.数据可实时读取；5.可直接测序RNA；

缺点：错误率目前相对较高，且是随机错误。

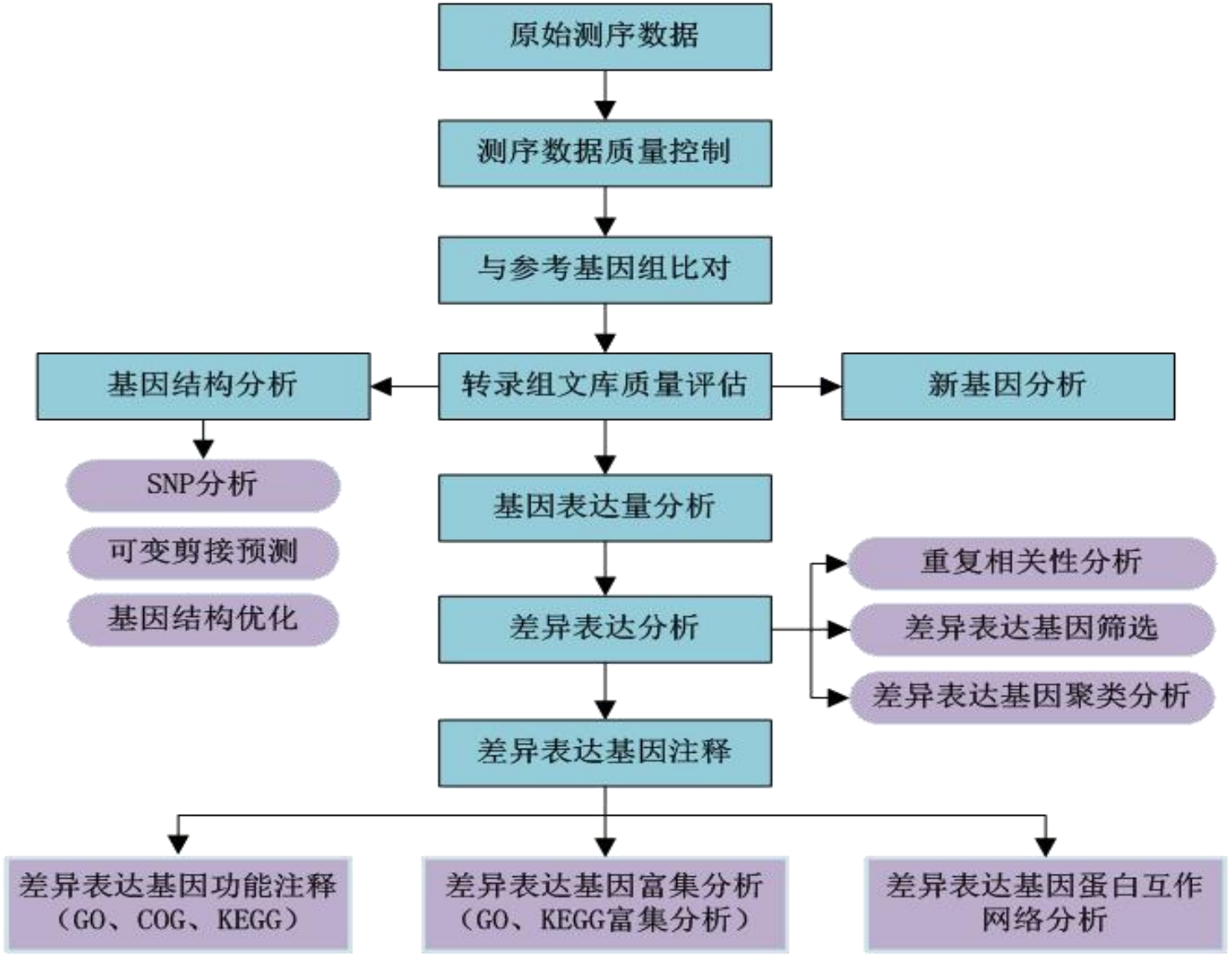
RNAseq测序原理介绍

实验流程



RNAseq测序原理介绍

生物信息分析流程



数据质量控制

测序数据（reads）格式：fastq

```
@ERR1457197.1.1 HWI-ST790:4:1101:1352:2084 length=100
CCGCAACAACCATTTCCCCAGGGATCAGAACAAATAATTCCCCAACAACCATTCCCTCTGCAACCACAACCATTCCCCCAACAACCACAACAACAATTGC
+ERR1457197.1.1 HWI-ST790:4:1101:1352:2084 length=100
CCCFFFFFHHHHHJJJJJJJJJJJJJJIIIJIIJJJJJJJJJJJDGIGIIBFHIGJJIJIGHE>ECEEDBB((;;>@C?BD(<B9?@<?A?7?<?+8@###
@ERR1457197.2.1 HWI-ST790:4:1101:1311:2101 length=100
GGCACATCTGTTAAAAGATAACGCAGGTGTCCTAAGATGAGCTCAACGAGAACAGAAATCTCGTGTGGAACAAAAGGGTAAAAGCTCGTTTGATTCTGAT
+ERR1457197.2.1 HWI-ST790:4:1101:1311:2101 length=100
BCCFFFFFHHHHHJJJJJJJJJJJIJJHIIJJIIIJGIIJJJIHJGGIIJGHJGCG@@FGGEGCEEHGFFFFDE@>B?=A>C;ABCA=;?BDA@CDA:>@
@ERR1457197.3.1 HWI-ST790:4:1101:1443:2109 length=100
GTTCTCGAACAGGTGCAAGTAGAAGGCAAGAGTTGCTGCAGGAGCACCCTAGGAAGAAACTGCTACAATCTTTGCCGCGTCCGTGGTGCTCAGAAGCTAT
+ERR1457197.3.1 HWI-ST790:4:1101:1443:2109 length=100
CCCFFFFFHHHHHEHIIJHHIJJJJJJJJJJJIJJJJJIIJJIIJJGHIJGIBFHJJCHJGGIGIIHHHGFFDFEB?CDBDDB?@<<BCA@CCC>?C@C#
```

FASTQ格式文件示意图

FASTQ文件中通常每4行对应一个序列单元：第一行以@开头，后面接着序列标识（ID）以及其它可选的描述信息；第二行为碱基序列，即Reads；第三行以"+"开头，后面接着可选的描述信息；第四行为Reads每个碱基对应的质量打分编码，长度必须和Reads的序列长度相同。

数据质量控制

下机后的测序数据需要进行质控和预处理，以保证后续分析所用的reads序列是干净可靠的。质控相关的软件有很多，例如cutadapt软件用于去除接头序列，trimmomatic软件可以根据碱基质量和reads长度过滤序列，fastqc软件可用于评估数据的质量，包括每个碱基的质量、类别，序列长度、K-mer序列、模糊碱基、冗杂序列和重复序列等。但这些软件大多不支持多线程，而且每个软件的功能有限，需要使用多个软件才能完成整个质控过程。目前使用最广泛的质控软件为fastp，其一次扫描分析就就可完成以上几款软件的所有功能，并且该软件支持多线程，运行速度非常快。

fastp软件的部分功能：

（1）过滤（包括低质量的序列、太短的序列、太多N的序列）；

（2）全局裁剪（在头/尾部，不影响去重），对于illumina下机数据往往最后一到两个cycle需要这样处理；

（3）去除接头，无需输入接头序列，算法会自动识别接头序列并进行去除；

（4）对于双端（PE）的数据，会自动查找每一对reads的重叠区域，并对该重叠区域中不匹配的碱基进行碱基进行校正。

（5）去除尾部的polyG，对于illumina，NovaSeq的测序数据，因为是两色法发光，polyG是常有的，所以该特性对这两类测序平台默认打开。

（6）对数据进行全方位质控，并生成人性化的质控报告。

数据质量控制

fastp命令示例：

```
fastp -i sampleA1_1.fastq.gz -I sampleA1_2.fastq.gz -o sampleA1.clean.R1.fq.gz -O sampleA1.clean.R2.fq.gz
```

参数说明：

-a, --adapter_sequence　　# 指定接头序列。（对于双端数据，通过read1和read2之间的overlap可判断接头，因此通常可以不指定接头序列）

--detect_adapter_for_pe　　# 开启双端的接头检测(应该不同于overlap方法)；

-5, --cut_front　　# 默认关闭，从5’开始滑动窗口，如果滑窗内的碱基质量值低于阈值，则切除，继续滑动，否则停止；

-3, --cut_tail　　# 默认关闭，从3’端开始滑动，同上；

-n, --n_base_limit　　# 限制N的数量；

--cut_window_size　　# 设置滑窗大小；

--cut_mean_quality　　# 平均质量值阈值；

-l, --length_required　　# 最小长度值；

-q, --qualified_quality_phred　　# 质量值低于此值认为是低质量碱基，默认15，表示质量为Q15；

数据质量控制

参数说明：

```
--json          # json格式的输出文件；
--html          # html格式的输出文件；
--failed_out    # 存储过滤的reads，过滤的原因会加入到read name中；
--unpaired1     # 存储read2被过滤但read1满足要求的read1；
--unpaired2     # 存储read1被过滤但read2满足要求的read2；
                # 这两个参数可指定相同的文件，来同时保存read1和read2
```

分析命令参考：

```
fastp -a auto --detect_adapter_for_pe -w 4 -i sampleA1_1.fastq.gz -I sampleA1_2.fastq.gz --cut_front --cut_tail --n_base_limit 5 --cut_window_size 4 --cut_mean_quality 20 --length_required 75 -o sampleA1.clean.R1.fq.gz -O sampleA1.clean.R2.fq.gz --json sampleA1.cleandata.fastp.json --html sampleA1.cleandata.fastp.html
```

数据质量控制

fastp质控结果报告：

fastp report

Summary

General

fastp version:	0.20.0 (https://github.com/OpenGene/fastp)
sequencing:	paired end (75 cycles + 75 cycles)
mean length before filtering:	75bp, 75bp
mean length after filtering:	75bp, 75bp
duplication rate:	0.673870%
Insert size peak:	119
Detected read1 adapter:	CCTGCCAGTAGCATATGCTTGTCTCAAAGATTAAGCCATGCATGTCTAAGTACGCACGGC
Detected read2 adapter:	CCTGCCAGTAGCATATGCTTGTCTCAAAGATTAAGCCATGCATGTCTAAGTACGCACGGC

Before filtering

total reads:	5.655738 M
total bases:	424.180350 M
Q20 bases:	402.707259 M (94.937745%)
Q30 bases:	373.008969 M (87.936409%)
GC content:	50.900847%

After filtering

total reads:	4.666578 M
total bases:	349.993350 M
Q20 bases:	344.505782 M (98.432094%)
Q30 bases:	326.456850 M (93.275158%)
GC content:	50.159394%

Filtering result

reads passed filters:	4.666578 M (82.510505%)
reads with low quality:	1.536000 K (0.027158%)
reads with too many N:	24 (0.000424%)
reads too short:	987.600000 K (17.461912%)

RPKM、FPKM、TPM的区别

1. Read count

概念：比对到gene A 的reads数

用途：（1）换算RPKM，FPKM等后续其他指标

（2）作为基因表达差异分析（DESeq，edgeR）的输入数据

2. RPKM

概念：Reads Per Kilobase of gene model per Million mapped reads

公式：

$$RPKM_g = \frac{\frac{r_g 10^3}{fl_g}}{\frac{R}{10^6}} = \frac{r_g \times 10^9}{fl_g \times R}$$

目标基因的数值：

r_g：每个基因的reads数；

fl_g：每个基因的长度；

参照的数值：

R：样本测序得到的总reads数

本质：

（1）以reads数为计数单位

（2）对基因长度（基因间的比较）和总数据量（样本间的比较）做矫正

RPKM、FPKM、TPM的区别

RPKM的优化：FPKM，TPM

3. FPKM

F=Fragment，即测序片段的数量。比对上的一对reads为一个fragment。

本质：以文库中的片段数量为计算单位，是在PE测序上对RPKM的校正。

4. TPM

T = Transcripts

公式：

$$TPM_g = \frac{\frac{r_g}{l_g} \times 1000000}{Sum(r_g/l_g + \dots + r_m/l_m)}$$

r_g：每个基因的reads数；
l_g：每个基因的长度;
$Sum(r_g/l_g + \dots + r_m/l_m)$：该样本所有校正后的reads数之和

TPM在一定条件下定量更准，尤其在样本间表达基因总数差异很大的情况下。

定量分析（hisat2+stringtie）

一. Hisat2软件

1. 简介

hisat2是一款短序列比对的工具，主要用于转录组数据的比对，是hisat比对工具的升级版。

2. 用法

（1）对参考基因组建立索引（hisat2-build）：

命令示例：`hisat2-build genome.fa genome.fa`

可选参数说明：

-p 线程数，根据计算机资源和参考基因组情况修改；

--large-index，4G以上的超大基因组建议加上这个参数；

索引建立完成后，会生成8个ht2结尾的文件，示例如右图：

```
genome.fa
genome.fa.1.ht2
genome.fa.2.ht2
genome.fa.3.ht2
genome.fa.4.ht2
genome.fa.5.ht2
genome.fa.6.ht2
genome.fa.7.ht2
genome.fa.8.ht2
```

定量分析（hisat2+stringtie）

2. 用法

（2）将质控后的reads序列比对至参考基因组上（hisat2）：

命令示例：`hisat2 –p 8 --dta –x genome.fa -1 sampleA1.R1.fastq.gz -2 sampleA1.R2.fastq.gz –S sampleA1.sam`

可选参数说明：

-p 线程数，根据计算机资源和参考基因组情况修改；

-x 参考基因组索引文件的前缀；

-1 R1端的reads序列；

-2 R2端的reads序列；

-S 输出文件名（sam格式）

定量分析（hisat2+stringtie）

二. stringtie软件

1. 简介

StringTie是一种快速高效的RNA-Seq序列比对组装器。它的输入为短序列的比对结果。为了在实验之间鉴定差异表达的基因，可以使用Ballgown，Cuffdiff或其他（DESeq，edgeR等）专用软件来处理StringTie的输出。

2. 用法

（1）转录本组装（stringtie）：

命令示例：`stringtie -p 8 -G genome.gtf -o sampleA1.gtf -l sampleA1 sampleA1.bam`

可选参数说明：

-p 线程数，根据计算机资源和参考基因组情况修改，默认1；

-G 指定基因组的gtf注释文件，在-e未设置的条件下，输出中包括注释文件中的转录本和预测的新型转录本；

-o 指定输出文件名，默认输出为标准输出；

-l 输出文件的前缀名；

-e 加上该参数表示只统计可以匹配-G指定的参考gtf中的转录本，不再对新的转录本做预测，这可以加快程序的运行速度；

定量分析（hisat2+stringtie）

2. 用法

（2）转录本合并（stringtie --merge）：

命令示例：stringtie --merge –p 8 –G genome.gtf -o stringtie_merge.gtf mergelist.txt

可选参数说明：

-p　线程数，根据计算机资源和参考基因组情况修改，默认1；

-G　指定基因组的gtf注释文件，在-e未设置的条件下，输出中包括注释文件中的转录本和预测的新型转录本；

-o　指定输出文件名，默认输出为标准输出；

-l　输出文件的前缀名；

定量分析（hisat2+stringtie）

2. 用法

（3）转录本（基因）的定量（stringtie）：

命令示例：`stringtie -B -p 8 -G stringtie_merge.gtf -o sampleA1.gtf sampleA1.bam`

可选参数说明：

-p 线程数，根据计算机资源和参考基因组情况修改，默认1；

-G 指定基因组的gtf注释文件，在-e未设置的条件下，输出中包括注释文件中的转录本和预测的新型转录本；

-o 指定输出文件名，默认输出为标准输出；

-b 指定 *.ctab 文件的输出路径, 而非-o选项指定的路径；

-B 使用该选项，则会输出Ballgown输入表文件（* .ctab）,在-b未设置的情况下，使用-o的路径输出；

定量分析（hisat2+stringtie）

三. samtools软件

1. 简介

Samtools是一个用来处理SAM/BAM格式的比对文件的工具，它能够输入和输出SAM/BAM格式的文件，对其进行排序、合并、建立索引等处理。排序后的BAM文件通常用于后续转录本的组装与定量。

2. 用法

对sam文件进行排序，并转换成bam格式文件：

命令示例：`samtools sort -@ 8 -o sampleA1.bam sampleA1.sam`

可选参数说明：

sort 排序命令

-@ 线程数，默认是单线程；

-o 设置最终排序后的输出文件名；

定量分析（hisat2+stringtie）

实操代码

```
# 基因组建索引
hisat2-build -p 4 ref/genome.fa ref/genome.fa
# hisat比对
hisat2 -p 4 --dta -x ref/genome.fa -1 result/1.QC/sampleA1/sampleA1.clean.R1.fq.gz -2
result/1.QC/sampleA1/sampleA1.clean.R2.fq.gz -S result/2.hisat/sampleA1/sampleA1.sam
# 对sam文件排序并转换成bam文件
samtools sort -@ 4 -o result/2.hisat/sampleA1/sampleA1.bam result/2.hisat/sampleA1/sampleA1.sam
# 转录本组装
stringtie -p 4 -G ref/genome.gtf -o result/3.stringtie/sampleA1/sampleA1.gtf -l sampleA1 result/2.hisat/sampleA1/sampleA1.bam
# 转录本合并
#（1）制作mergelist.txt文件
ls result/3.stringtie/*/* > result/3.stringtie/merge/mergelist.txt
#（2）转录本合并
stringtie --merge -p 4 -G ref/genome.gtf -o result/3.stringtie/merge/stringtie_merge.gtf result/3.stringtie/merge/mergelist.txt
# 转录本定量
stringtie -b result/4.ballgown/data/sampleA1/ -p 4 -G result/3.stringtie/merge/stringtie_merge.gtf -o
result/3.stringtie/quantify/sampleA1.gtf result/2.hisat/sampleA1/sampleA1.bam
```

差异分析（Ballgown）

Ballgown是一个专门用来进行差异分析的R包，它可以直接读入stringtie的转录组组装和定量数据。

分析代码:

1. 加载相关R包，读取数据

```
# 安装R包
BiocManager::install("ballgown")
# 加载R包
library(ballgown)
library(genefilter)
library(dplyr)
# 读入样本的分组信息
pheno_data = read.csv("group.csv",sep="\t",header=TRUE)
# 读入定量分析数据
bg_data = ballgown(dataDir="result/4.ballgown/data/",samplePattern="sample",pData=pheno_data)
```

差异分析（Ballgown）

2. 数据筛选

```
# 筛选FPKM在样本中方差大于1的转录本
bg_filt = subset(bg_data,"rowVars(texpr(bg_data))>1",genomesubset=TRUE)
```

3. 差异表达分析

```
# 对转录本进行差异分析
results_transcripts=stattest(bg_filt,feature="transcript",covariate="group",getFC=TRUE,meas="FPKM")
# 对基因进行差异分析
results_genes=stattest(bg_filt,feature="gene",covariate="group",getFC=TRUE,meas="FPKM")
# 给转录本添加基因名和基因ID
results_transcripts=data.frame(geneNames=ballgown::geneNames(bg_filt),geneIDs=ballgown::geneIDs(bg_filt),results_transcripts)
```

差异分析（Ballgown）

4. 差异分析结果处理

```
# 根据p-value值对分析结果进行排序(由小到大)
results_transcripts = arrange(results_transcripts,pval)
results_genes = arrange(results_genes,pval)
# 提取显著差异表达的基因和转录本（即q value值小于0.05）
results_transcripts_0.05 = subset(results_transcripts,results_transcripts$qval < 0.05)
results_genes_0.05 = subset(results_genes,results_genes$qval < 0.05)
# 将显著差异表达的基因和转录本写入输出文件
write.table(results_transcripts_0.05,file="result/4.ballgown/transcripts_0.05.xls",sep="\t",quote=FALSE)
write.table(results_genes_0.05,file="result/4.ballgown/genes_0.05.xls",sep="\t",quote=FALSE)
```

差异分析（Ballgown）

Ballgown分析结果展示

A1 feature

	A	B	C	D	E	F
1	feature	id	fc	pval	qval	
2	gene	MSTRG.524	0.002372333	2.35E-11	2.39E-08	
3	gene	MSTRG.139	3.12379154	2.34E-06	0.001188229	
4	gene	MSTRG.523	0.082472824	7.75E-06	0.002625788	
5	gene	MSTRG.609	7.269403794	1.19E-05	0.003014923	
6	gene	MSTRG.613	9.080527523	5.18E-05	0.010527516	
7	gene	MSTRG.370	0.62487256	7.30E-05	0.012365126	
8	gene	MSTRG.512	0.638579154	8.56E-05	0.012434225	
9	gene	MSTRG.759	0.154662154	0.000165343	0.021019207	
10	gene	MSTRG.60	0.61556077	0.000251731	0.02844563	
11	gene	MSTRG.760	0.124063884	0.000301255	0.03063765	
12	gene	MSTRG.615	7.807480718	0.000395587	0.036573845	
13	gene	MSTRG.228	1.41110184	0.00045895	0.038895983	
14	gene	MSTRG.23	4.021481965	0.00062462	0.048864533	
15						

fc：差异倍数

pval：原始的p值

qval：校正后的p值

差异分析（DESeq2）

DESeq2是DESeq的更新版本。两者分析原理相同，但DESeq2使用更简单。

两者最大的区别在于：DSEeq使用estimateSizeFactors()函数和estimateDispersions()函数分别估计大小因子和方差，而DESeq2使用DESeq()函数一步完成大小因子和方差的估计。

DESeq

```
> cds <- estimateSizeFactors(cds)
> sizeFactors(cds)
       A1        A2        A3        B1        B2        B3
0.2443908 0.2443908 2.2807146 2.2807146 1.8702027 1.8702027
```

```
> cds <- estimateDispersions(cds)
> cds
CountDataSet (storageMode: environment)
assayData: 24791 features, 6 samples
  element names: counts
protocolData: none
phenoData
  sampleNames: A1 A2 ... B3 (6 total)
  varLabels: sizeFactor condition
  varMetadata: labelDescription
featureData
  featureNames: 1 2 ... 24791 (24791 total)
  fvarLabels: disp_pooled
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

DESeq2

```
> dds <- DESeq(dds,parallel = T)
estimating size factors
estimating dispersions
gene-wise dispersion estimates: 2 workers
mean-dispersion relationship
final dispersion estimates, fitting model and testing: 2 workers
```

差异分析（DESeq2）

DESeq2差异分析的三个主要数据：

- 表达矩阵（原始的count数据）
- 分组矩阵（实验设计的分组情况）
- 差异比较矩阵（指定差异比较对象）

```
> head(countsTable)
          sample1 sample2 sample3 sample4 sample5 sample6 sample7
NM_000032       0       0       0       0       0       0       0
NM_000033      82     116      47     147      52      79     117
NM_000044       0       0       0       0       0       0       1
NM_000047       0       0       0       0       0       0       0
NM_000052       6       1       4       3       5       7       9
NM_000054       0       0       0       0       0       0       0
```

```
> countsDesign <- data.frame(row.names = colnames(countsTable), group
 = c("FY","FG","FY","FG","FG","FY","MY","MY","MY","MG","MG","MG"))
> head(countsDesign)
        group
sample1    FY
sample2    FG
sample3    FY
sample4    FG
sample5    FG
sample6    FY
```

```
> population_res <- results(dds,contrast = c("group","FY","FG"), para
llel=T)
```

差异分析（DESeq2）

DESeq2差异分析的三个主要步骤：

1. DESeqDataSetFromMatrix()构造dds对象

```
> dds <- DESeqDataSetFromMatrix(countsMatrix,colData = countsDesign,
 design = ~0 + sex+population)
> dds
class: DESeqDataSet
dim: 2082 12
metadata(1): version
assays(1): counts
rownames(2082): NM_000032 NM_000033
  ... NR_132964 NR_133641
rowData names(0):
colnames(12): sample1 sample2 ...
  sample11 sample12
colData names(2): sex population
```

2. DESeq()函数进行差异分析

```
> dds <- DESeq(dds,parallel = T)
estimating size factors
estimating dispersions
gene-wise dispersion estimates: 2 workers
mean-dispersion relationship
final dispersion estimates, fitting model and testing: 2 workers
```

3. results()函数提取差异分析结果

```
> population_res <- results(dds,contrast = c("population","YRI","GBR"), parallel=T)
> head(population_res)
log2 fold change (MLE): population YRI vs GBR
Wald test p-value: populationYRI
DataFrame with 6 rows and 6 columns
             baseMean log2FoldChange     lfcSE       stat     pvalue      padj
            <numeric>      <numeric> <numeric>  <numeric>  <numeric> <numeric>
NM_000032   0.07996103      0.4223028 3.1165321  0.1355041 0.89221335        NA
NM_000033  86.95610894     -0.4512666 0.2563390 -1.7604288 0.07833513 0.4831861
NM_000044   0.06492779      0.4223028 3.1165321  0.1355041 0.89221335        NA
NM_000052   4.77650437      1.1472105 0.5739904  1.9986579 0.04564538 0.3850088
NM_000074   0.28266537      0.3930486 2.8338500  0.1386977 0.88968900        NA
NM_000117 662.38720398     -0.3647416 0.1863547 -1.9572445 0.05031872 0.3904733
```

差异分析（DESeq2）

分析前数据准备

DESeq2分析的输入文件是reads的计数文件，该文件可从比对后的sam文件中产生。

软件：htseq-count

命令：

```
samtools sort -n -@ 8 -o ERR188044_chrX.sam ERR188044_chrX.bam
```

```
htseq-count  ERR188044_chrX.sam  chrX.gtf  > ERR188044_chrX_counts.txt
```

差异分析（DESeq2）

输入数据格式

gene_id	sampleA1	sampleA2	sampleA3	sampleB1	sampleB2	sampleB3
A1BG	46	83	47	57	61	59
A1CF	0	0	0	0	0	0
A2M	8	8	10	4	10	6
A2ML1	0	1	0	0	0	0
A3GALT2	131	57	135	96	93	93
A4GALT	0	1	0	0	0	0
A4GNT	0	0	0	0	0	0
AAAS	433	540	445	534	378	575
AACS	233	266	216	291	232	316
AADAC	0	0	0	0	0	0
AADACL2	0	0	0	0	0	0
AADACL3	0	0	0	0	0	0
AADACL4	0	0	0	0	0	0
AADAT	0	1	0	0	4	0
AAED1	173	258	180	179	141	195
AAGAB	1233	1410	1215	1154	914	1174
AAK1	6875	7005	6852	8978	7627	9087
AAMDC	86	96	81	63	84	64
AAMP	1215	1329	1243	1219	1017	1320
AANAT	0	7	0	0	5	0
AAR2	527	600	549	607	501	628
AARD	0	0	0	4	1	4
AARS	834	1092	859	1150	888	1250
AARS2	774	786	786	866	643	896

差异分析（DESeq2）

分析代码:

1. 加载相关R包，读取数据

```
# 安装R包
BiocManager::install("DESeq2")
# 加载R包
library(DESeq2)
# 读取count数据表
countsTable <- read.delim("result/5.DESeq2/counts.xls",header = T,row.names = 1)
# 读取样本分组信息
countsDesign <- read.csv("group.csv",sep="\t",header=TRUE)
```

差异分析（DESeq2）

2. 数据转换处理

```
# 先将读取数据由data.frame转换成matrix类型
countsMatrix <- as.matrix(countsTable)
# 再将countsMatrix转换成DESeq2的数据格式（构建dds）：
dds <- DESeqDataSetFromMatrix(countsMatrix,colData = countsDesign, design = ~ group)
```

3. 数据过滤及差异分析

```
# 将所有样本中count值均为0的基因去除
dds <- dds[rowSums(counts(dds))>0,]
# DESeq差异分析：大小因子的估计，离差的估计，负二项分布的拟合以及计算相应的统计量
dds <- DESeq(dds,parallel = T)
# 提取组间差异表达基因的分析结果：
group_res <- results(dds, contrast = c("group","A","B"), parallel=T)
```

差异分析（DESeq2）

4. 差异分析结果处理

```
# 提取组间显著差异表达的基因（padj<0.01）：
group_Sig_0.01 <- group_res[which(group_res$padj<0.01),]
# 将分析结果写入文件：
write.table(group_Sig_0.01,file = "result/5.DESeq2/group_Sig_0.01_DESeq2.xls",sep="\t",quote=FALSE)
```

差异分析（DESeq2）

DESeq2分析结果展示

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
NM_001007	23450.95647	0.855609742	0.164375844	5.205203657	1.94E-07	1.20E-05
NM_001415	5460.507266	0.501891883	0.11537103	4.350241853	1.36E-05	0.000503154
NM_005044	795.8099816	0.540185252	0.152487858	3.542480435	0.000396383	0.008147869
NM_005089	239.9765864	0.575319942	0.148417982	3.876349321	0.000106035	0.002930439
NM_005765	1023.274913	-0.417323127	0.108908839	-3.831857281	0.00012718	0.002941026
NM_145799	1393.872872	0.643813187	0.144813095	4.445821612	8.76E-06	0.000404949
NR_001564	2479.443797	9.200250215	0.366758906	25.08528103	7.20E-139	6.66E-137
NR_003255	1614.435793	8.852891084	0.302744313	29.24213832	5.65E-188	1.05E-185
NR_024582	43.37659388	0.873009241	0.225848917	3.865456838	0.000110881	0.002930439

baseMean：经过矫正的reads count均值；

log2FoldChange：对差异倍数取以2为底的对数；

lfcSE：差异倍数取对数后的标准差；

stat：Wald统计量，由logfoldchange除以标准差所得；

pvalue：原始的p值

padj：校正后的p值

差异分析（edgeR）

edgeR差异分析的三个主要数据：

- 表达矩阵（原始的count数据）
- 分组矩阵（实验设计的分组情况）
- 差异比较矩阵（指定差异比较对象）

```
> rawdata <- read.delim("DE_data.tsv",header = F,row.names = 1)
> sampleNames <- paste(c("sample"),1:12,sep = "")
> colnames(rawdata) <- sampleNames
> head(rawdata)
          sample1 sample2 sample3 sample4 sample5 sample6 sample7 sample8
NM_000032       0       0       0       0       0       0       0       0
NM_000033      82     116      47     147      52      79     117      49
NM_000044       0       0       0       0       0       0       1       0
NM_000047       0       0       0       0       0       0       0       0
NM_000052       6       1       4       3       5       7       9      10
NM_000054       0       0       0       0       0       0       0       0
```

```
> countsDesign <- data.frame(row.names = colnames(rawdata), group = c("FY","FG","FY"
,"FG","FG","FY","MY","MY","MY","MG","MG","MG"))
> head(countsDesign)
        group
sample1    FY
sample2    FG
sample3    FY
sample4    FG
sample5    FG
sample6    FY
```

```
> my.contrasts <- makeContrasts(YRI.f.vs.m = FY-MY, GBR.f.vs.m = FG-MG, F.
YRI.vs.GBR = FY-FG, M.YRI.vs.GBR = MY-MG,levels = design)
> my.contrasts
      Contrasts
Levels YRI.f.vs.m GBR.f.vs.m F.YRI.vs.GBR M.YRI.vs.GBR
    FG          0          1          -1           0
    FY          1          0           1           0
    MG          0         -1           0          -1
    MY         -1          0           0           1
> qlf_YRI_f.vs.m <- glmQLFTest(fit,contrast = my.contrasts[,"YRI.f.vs.m"])
```

差异分析（edgeR）

edgeR差异分析的五个主要步骤：

1. DGEList()构造DGEList对象

```
> data <- DGEList(counts = rawdata, group = countsDesign$group)
> data
An object of class "DGEList"
$counts
         sample1 sample2 sample3 sample4 sample5 sample6 sample7
NM_000032       0       0       0       0       0       0       0
NM_000033      82     116      47     147      52      79     117
NM_000044       0       0       0       0       0       0       1
NM_000047       0       0       0       0       0       0       0
NM_000052       6       1       4       3       5       7       9
         sample8 sample9 sample10 sample11 sample12
NM_000032       0       1        0        0        0
NM_000033      49      88      116      102       76
```

2. model.matrix()函数构建设计矩阵

```
> design <- model.matrix(~0 + group,data = countsDesign)
> colnames(design) <- levels(countsDesign$group)
> head(design)
        FG FY MG MY
sample1  0  1  0  0
sample2  1  0  0  0
sample3  0  1  0  0
sample4  1  0  0  0
sample5  1  0  0  0
sample6  0  1  0  0
```

差异分析（edgeR）

edgeR差异分析的五个主要步骤：

3. estimateDisp()函数进行差异分析

```
> data <- estimateDisp(data,design)
> data
An object of class "DGEList"
$counts
          sample1 sample2 sample3 sample4 sample5 sample6 sample7 sample8
NM_000033      82     116      47     147      52      79     117      49
NM_000052       6       1       4       3       5       7       9      10
NM_000074       1       0       1       0       0       0       0       0
NM_000117     685     790     302     939     352     763     756     630
NM_000132       3       1       5      15       8       2       6       4
          sample9 sample10 sample11 sample12
NM_000033      88      116      102       76
NM_000052       6        5        3        1
NM_000074       0        1        0        0
NM_000117     599      979      741      737
```

4. glmQLFit()函数构建DGEGLM对象

```
> fit <- glmQLFit(data,design)
> fit
An object of class "DGEGLM"
$coefficients
                  FG          FY          MG          MY
NM_000033  -7.802479  -8.185525  -7.961915  -8.212857
NM_000052 -11.297698 -10.680326 -11.433129 -10.527701
NM_000074 -14.617819 -12.687822 -13.344240 -14.617819
NM_000117  -5.907327  -6.097569  -5.839304  -6.162793
NM_000132 -10.425764 -11.118171 -10.694221 -11.438790
511 more rows ...

$fitted.values
              sample1    sample2     sample3    sample4    sample5
NM_000033  98.7765352  83.587420  38.9052998 144.880885  86.875245
```

差异分析（edgeR）

edgeR差异分析的五个主要步骤：

5. glmQLFTest()函数进行QL F-test

```
> qlf_YRI_f.vs.m <- glmQLFTest(fit,contrast = my.contrasts[,"YRI.f.vs.m"])
> qlf_YRI_f.vs.m
An object of class "DGELRT"
$coefficients
                   FG          FY          MG          MY
NM_000033  -7.802479  -8.185525  -7.961915  -8.212857
NM_000052 -11.297698 -10.680326 -11.433129 -10.527701
NM_000074 -14.617819 -12.687822 -13.344240 -14.617819
NM_000117  -5.907327  -6.097569  -5.839304  -6.162793
NM_000132 -10.425764 -11.118171 -10.694221 -11.438790
511 more rows ...

$fitted.values
```

edgeR的输入数据格式同DESeq2

差异分析（edgeR）

分析代码:

1. 加载相关R包，读取数据

```
# 安装R包
BiocManager::install("edgeR")
# 加载R包
library(limma)
library(edgeR)
# 读取count数据表
countsTable <- read.delim("result/5.DESeq2/counts.xls",header = T,row.names = 1)
# 读取样本分组信息
countsDesign <- read.csv("group.csv",sep="\t",header=TRUE)
```

差异分析（edgeR）

2. 数据处理

```
# 建立DGEList对象(edgeR的数据分析格式)
dg_list <- DGEList(counts = countsTable, group = countsDesign$group)
# 过滤至少有3个样本的counts数大于1
keep <- rowSums(cpm(dg_list)>1) >= 3
dg_list <- dg_list[keep,,keep.lib.sizes=FALSE]
# 数据标准化
dg_list <- calcNormFactors(dg_list)
# 建立设计矩阵
design <- model.matrix(~0 + group,data = countsDesign)
# 修改设计矩阵的列名
colnames(design) <- levels(factor(countsDesign$group))
```

差异分析（edgeR）

3. 差异分析

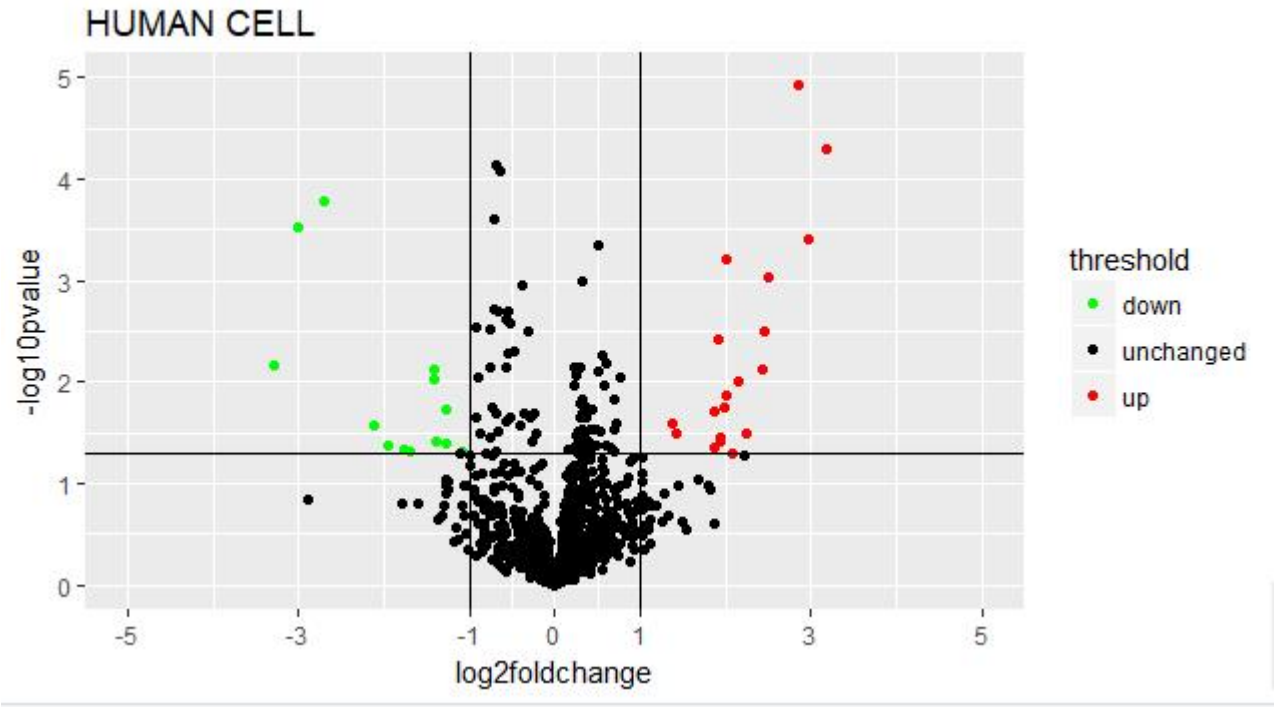
```
# 估计方差
dg_list <- estimateDisp(dg_list,design)
# 构建DGEGLM对象
fit <- glmQLFit(dg_list,design)
#建立对照：
my.contrasts <- makeContrasts(A.vs.B = A - B, levels = design)
# 差异分析（用 quasi-likelihood (QL) F-test ）
qlf_A.vs.B <- glmQLFTest(fit,contrast = my.contrasts[,"A.vs.B"])
qlf_A.vs.C <- glmQLFTest(fit,contrast = my.contrasts[,"A.vs.C"])
```

差异分析（edgeR）

4. 差异分析结果处理

```
# 对每组差异分析结果的pvalue进行校正
A.vs.B_padjust <- p.adjust(qlf_A.vs.B$table$PValue, method = "BH")
A.vs.C_padjust <- p.adjust(qlf_A.vs.C$table$PValue, method = "BH")
# 提取显著差异表达的基因
Sig_A.vs.B <- qlf_A.vs.B$table[which(A.vs.B_padjust < 0.05),]
# 将分析结果写入文件
write.table(Sig_A.vs.B,file = "result/6.edgeR/Sig_A.vs.B_0.05_edgeR.xls",sep="\t",quote=
FALSE,row.names=T)
```

火山图绘制



火山图(Volcano plot)常用于展示显著差异表达的基因，图中的x轴是特征值取log2转换的倍数变化值，该值大于0表示上调基因，小于0表示下调基因，越偏离中心差异倍数越大；y轴是取-log10变换的p值或者调整后的p值，值越大差异越显著。

火山图绘制

绘图**R**包 ggplot2

绘图数据示例

Names	log2FoldChange	p-value
A1BG	0.153785944	1
A1CF	-0.974770844	1
A2M	-5.888962627	0.421417793
A2ML1	5.974342187	0.996093185
A4GALT	1.9746723	0.41106371
AAAS	-0.140525778	0.106501868
AACS	0.094529455	0.200905357
AADAC	0.889422086	0.332491269
AADAT	-0.547154227	0.000157441
AAED1	0.507783549	3.43E-08
AAGAB	0.837890702	5.84E-43
AAK1	0.561546417	1.83E-11
AAMDC	-0.180741332	0.271400932
AAMP	-0.073098679	0.190075508
AAR2	-0.128046537	0.070945419
AARS	-1.586350138	6.01E-155
AARS2	0.068399226	0.445028567

火山图绘制

分析代码:

1. 加载相关R包，读取数据

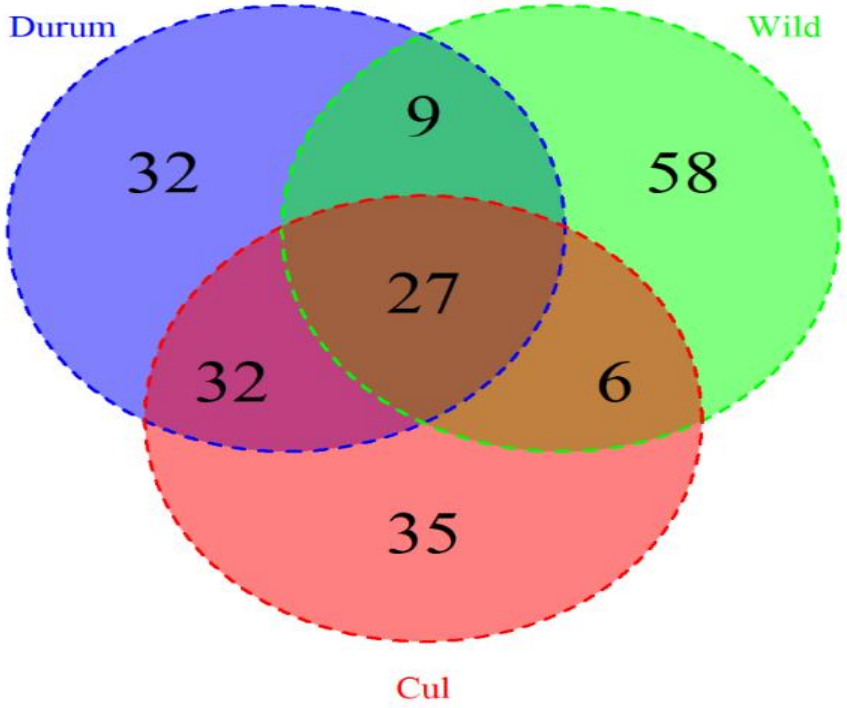
```
# 加载R包
library(ggplot2)
# 读取绘图数据
plotdata <- read.csv("result/7.plot_volcano/volcano_data.xls",sep="\t",header=TRUE)
# 数据分组处理
plotdata$change <- ifelse(plotdata$p_value<0.00000001 &
abs(plotdata$log2FoldChange)>=1,ifelse(plotdata$log2FoldChange>1,"Up","Down"),"Unchange")
```

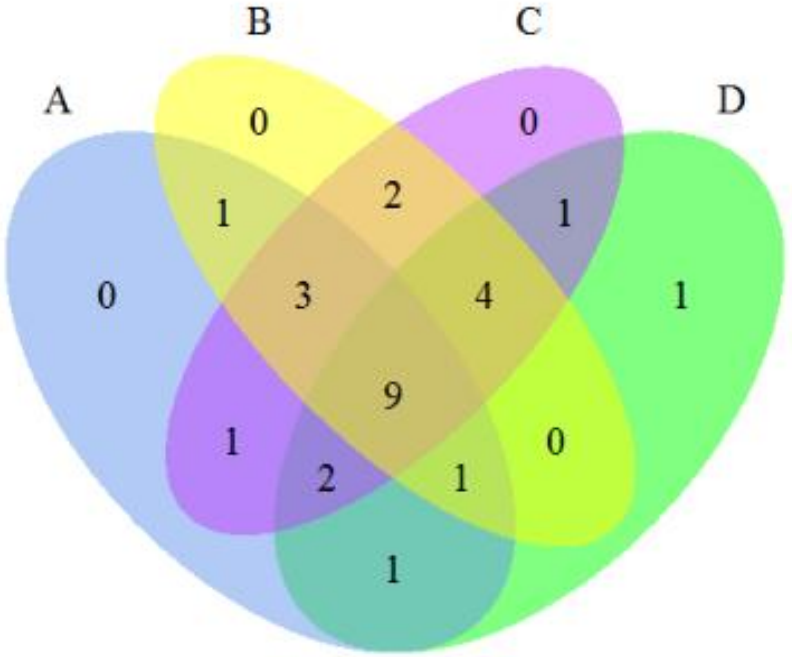
火山图绘制

2. 绘图

```
ggplot(plotdata, aes(x=log2FoldChange, y=-log10(p-value), colour = gene)
+ geom_point()                    # 绘制散点图
+ xlab("log2foldchange")          # 指定X轴坐标名称
+ ylab("-log10pvalue")            # 指定Y轴坐标名称
+ scale_x_continuous(limits = c(-10,10), breaks = c(-5,-3,-1,0,1,3,5))    # 指定X坐标轴范围和刻度线
+ ylim(0,300)                        # 指定Y坐标轴范围
+ scale_colour_manual(values = c("green","black", "red"), labels = c("down", "unchanged", "up"))
+ ggtitle("volcano plot")               # 指定图片主题名
+ geom_hline(yintercept = 1.3)         # 指定Y轴阈值线（即-log10(0.05)的值）
+ geom_vline(xintercept = c(-1,1)))    # 指定X轴阈值线（即-log2(-2)和-log2(2)的值）
```

venn图绘制





韦恩图（Venn Diagram）主要用来展示多个集合之间的逻辑关系

venn图绘制

绘图包　VennDiagram

绘图数据示例

A	B	C
g1	g1	g1
g2	g2	g2
g3	g3	g5
g4	g7	g7
g5	g8	g10
g6	g9	g13
g7	g10	g16
g8	g11	g19
g9	g12	g22
g10		
g11		

venn图绘制

几种绘图函数介绍

- 各个集合内的元素未知，只知道各个集合及相互之间交集的大小

```
draw.pairwise.venn()          #绘制两个集合的venn图
draw.triple.venn()            #绘制三个集合的venn图
draw.quad.venn()              #绘制四个集合的venn图
draw.quintuple.venn()         #绘制五个集合的venn图
```

- 知道各个集合内的元素

```
venn.diagram()                #可绘制2-5个集合的venn图
```

venn图绘制

venn.diagram函数及默认参数

venn.diagram(x, filename, height = 3000, width = 3000, resolution = 500, lwd = 2, lty = 2, col = "black", fill = NULL, alpha = 0.5, rotation.degree = 30, rotation.centre = c(20,40), rotation = 1, cat.pos = c(-40,40,180), cat.cex = 1, cat.col = "black", cat.dist = c(0.05,0.05,0.025), imagetype = "tiff", na = "stop", main = NULL, sub = NULL, main.pos = c(0.5, 1.05), main.fontface = "plain", main.fontfamily = "serif", main.col = "black", main.cex = 1, main.just = c(0.5, 1), sub.pos = c(0.5, 1.05), sub.fontface = "plain", sub.fontfamily = "serif", sub.col = "black", sub.cex = 1, sub.just = c(0.5, 1), category.names = names(x))

venn图绘制

分析代码:

1. 加载相关R包，读取数据

```
# 加载R包
library(VennDiagram)
# 读取绘图数据
plotdata <- read.csv("8.plot_venn/venn_data.xls",sep="\t",header=TRUE)
```

venn图绘制

2. 绘图（以绘制3个集合的venn图为例）

（1）已知每个集合的具体元素，venn.diagram()函数作图

venn.diagram(plotdata, height = 1000,width = 1000, col = c("green","red","blue"),fill = c("green","red","blue"),lwd = 0.8, lty = 2, cat.dist = 0.05,cat.col = c("green","red","blue"),cat.cex = 0.5,category.names = c("Wild","Cul","Durum"), filename = "data.png", imagetype = "png")

cex：数字的字体大小；alpha：透明度；fontface：字体粗细（加粗bold）；col：圆圈线条颜色；fill：圆圈填充颜色；lwd：圆圈线条粗细；lty：圆圈线条形状（1实线，2虚线，blank无线条）；cat.dist：标签距离圆圈的远近；cat.col：标签的颜色；cat.cex：标签字体大小；category.names：各集合的标签名；filename：输出文件名； imagetype：输出图片的格式类型。

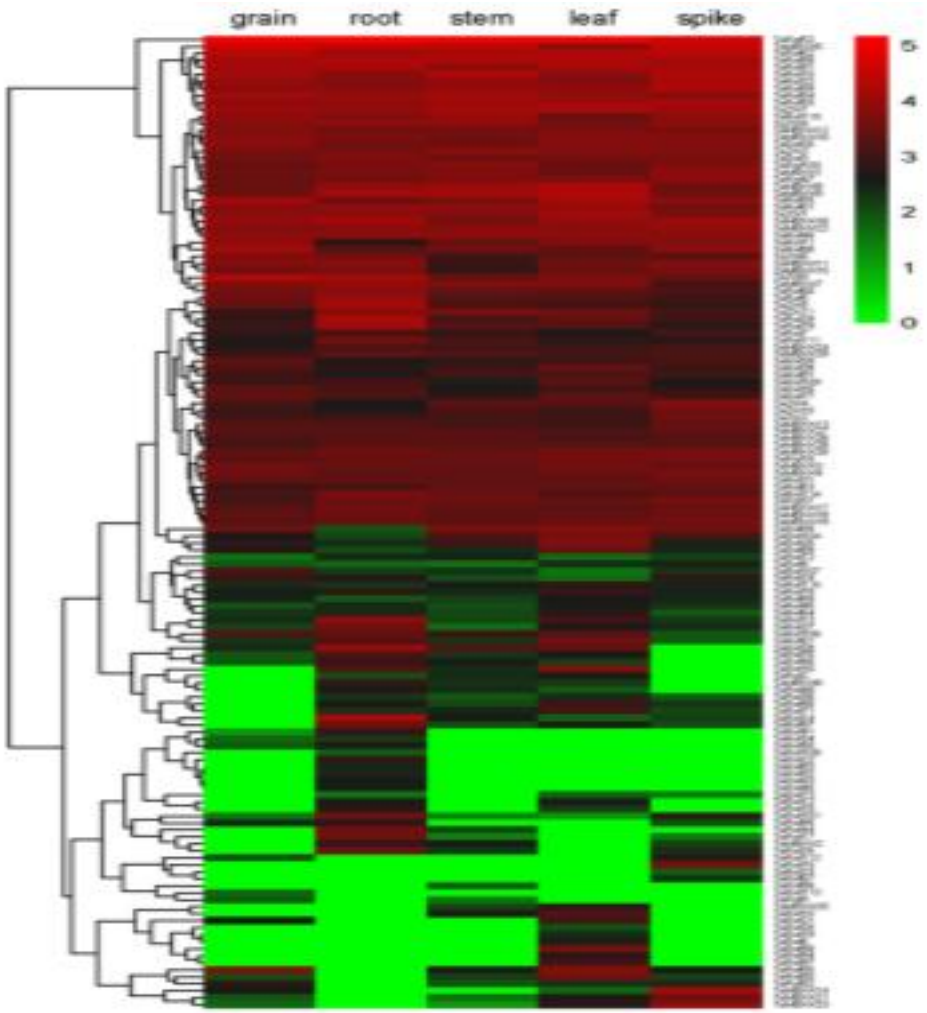
venn图绘制

2. 绘图（以绘制**3**个集合的**venn**图为例）

（**1**）每个集合的具体元素未知，但各集合交集的大小已知，**draw.triple.venn()**函数作图

```
A <- length(venn_data$V1)
B <- length(venn_data$V2)
C <- length(venn_data$V3)
AB <- length(intersect(venn_data$V1,venn_data$V2))
AC <- length(intersect(venn_data$V1,venn_data$V3))
BC <- length(intersect(venn_data$V2,venn_data$V3))
ABC <- length(intersect(intersect(venn_data$V1,venn_data$V2),venn_data$V3))
draw.triple.venn(area1 = A,area2 = B,area3 = C,n12 = AB,n13 = AC,n23 = BC,n123 = ABC,category = c("Wild","Cul","Durum"),col = c("red","yellow","blue"),fill = c("red","yellow","blue"),lwd = 0.8,cat.cex = 1.5,cat.col = c("red","yellow","blue"),cat.dist = 0.08,lty = 2)
```

热图绘制



热图通常用来展示两组或多组处理中特定基因（或外显子）的表达情况。热图使用不同颜色尺度来呈现基因（或外显子等）的表达量。图中的特征值按列（样本）和行（基因）排列。每个图形单元的颜色是由表达值的大小决定的。同时会对列（样本）或行（基因）进行分层聚类，聚类结果会在图的上部或左部以树的形式展现。

热图绘制

绘图包　pheatmap

绘图数据示例（每个样本基因表达量的**fpkm**值）

gene_id	Sample_a1	Sample_a2	Sample_a3	Sample_b1	Sample_b2	Sample_b3
gene1	9.09863	8.39823	8.78762	3.93766	8.25665	2.27406
gene2	4.81758	6.51911	4.00972	2.50362	2.37066	2.55658
gene3	0.364478	1.21141	1.2551	3.15624	6.3716	4.27103
gene4	0.606147	0.470184	0.410991	4.28929	2.31695	4.4292
gene5	18.8357	24.117	20.0295	130	24.8569	113.384
gene6	3.49204	6.34534	7.70006	57.0906	49.9272	65.6909
gene7	11.9717	16.0106	17.8544	117.289	62.6742	123.859
gene8	32.7282	33.4583	26.2566	8.97169	7.32371	6.66194
gene9	4.35666	3.96312	4.80725	1.41522	0.746992	1.60811
gene10	2.86508	3.16508	3.04327	11.0899	8.70267	14.1976
gene11	5.86777	4.68516	4.1543	1.06214	1.39622	1.41194
gene12	0.678623	1.43174	1.30874	4.67139	3.84297	4.88139
gene13	77.6258	41.2245	28.729	7.22683	9.98216	2.9473
gene14	233.686	169.447	131.682	5.77286	10.0183	5.2036
gene15	15.4462	17.3215	15.0502	2.53805	3.07854	1.60464
gene16	3.49646	3.75484	4.00898	15.3349	15.9494	21.051
gene17	10.224	8.37964	10.9951	4.16836	8.29093	4.77685
gene18	6.21501	6.4082	4.97029	3.02133	2.81837	2.51794
gene19	0.702188	1.26902	0.738406	5.59151	3.84197	6.59731
gene20	0.584959	0.429363	0.804435	2.86098	2.86587	4.05461
gene21	3.74586	6.2001	3.48496	85.0382	10.1351	85.7759
gene22	4.56035	7.36665	6.75567	29.6468	30.4706	42.4688
gene23	8.61111	11.1033	10.3717	71.1556	38.7673	75.2048

热图绘制

pheatmap函数及默认参数

```
pheatmap(data, cluster_rows = TRUE, cluster_cols = TRUE, color = colorRampPalette(rev(brewer.pal(n=7,name= "RdYlBu")))(100), border_color = "grey60", scale = "none", cellwidth = NA, cellheight = NA, treeheight_row=50, treeheight_col=50, cutree_rows = NA, cutree_cols = NA, display_numbers = F, fontsize = 10 ,fontsize_number = 0.8 * fontsize， number_color = "grey30" ，show_rownames = T, show_colnames = T, main = NA， legend = TRUE, legend_labels = NA, annotation = NA, annotation_names_row = TRUE, annotation_names_col = TRUE)
```

热图绘制

分析代码:

1. 加载相关R包，读取数据

```
# 加载R包
library(pheatmap)
# 读取绘图数据
plotdata <- read.delim("result/9.plot_heatmap/heatmap_data.xls",header = T, row.name = 1)
```

热图绘制

2. 绘图

pheatmap(plotdata, color = colorRampPalette(c("blue","green","red"),bias = 1.2)(100), cellheight = 8, cellwidth = 35, cluster_cols = FALSE, scale = "column", cutree_cols = 4, cutree_rows = 2, display_numbers = T, number_color = "grey10", main = "DE in group", show_rownames = F)

cellheight：单元格高度；cellwidth：单元格宽度；cluster_cols：是否对列进行聚类；scale：表示进行均一化的方向（值为 "row", "column" 或者"none"）；cutree_cols：若进行了列聚类，根据列聚类数量分隔热图列；display_numbers：是否显示数字；number_color：数字颜色；main：热图的标题名；show_rownames：是否显示行名；

WGCNA分析

WGCNA：Weighted Gene Co-Expression Network Analysis

中文名：加权基因共表达网络分析

该分析旨在寻找协同表达的基因模块(module)，探索基因网络与关注的表型之间的关系，以及网络中的核心基因（hub基因）的鉴定。其中网络中点代表基因，边代表基因表达相关性。基于网络的基因筛选法，可用于识别候选生物标志物或治疗靶标。

WGCNA适用于复杂的数据模式，推荐15个样本以上的数据。

常见的应用方向：

- 不同器官/组织类型发育调控;
- 同一组织不同时期发育调控;
- 不同处理方式的作用机制;
- 病原体感染宿主的响应机制;

WGCNA分析

为什么要使用WGCNA分析?

- 差异分析标准的主观性太强;
- 大样本量情况下，差异分析无法对基因进行有效聚类;
- 强调基因集（module）而非单个基因;

WGCNA分析的主要内容

从方法上来讲，WGCNA分析分为表达量聚类分析和表型关联分析两部分。具体包含以下5个内容:

1. 共表达基因网络构建;
2. 基因模块鉴定;
3. 模块与性状的相关性;
4. 模块间的关系;
5. 模块内hub基因鉴定;

雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA分析数据准备

数据对象

- 表达量矩阵：转录组的表达量矩阵，RPKM、FPKM、TPM均可，要求基因在行，样本在列；
- 性状矩阵：用于关联分析的表型数据，必须是数值型特征，质量性状可以转换为0-1的矩阵形式；

样本数

- 至少15个样本

过滤

- top表达基因（如top 5000）
- 高变异系数基因（如最高的50%的基因）
- 差异表达基因

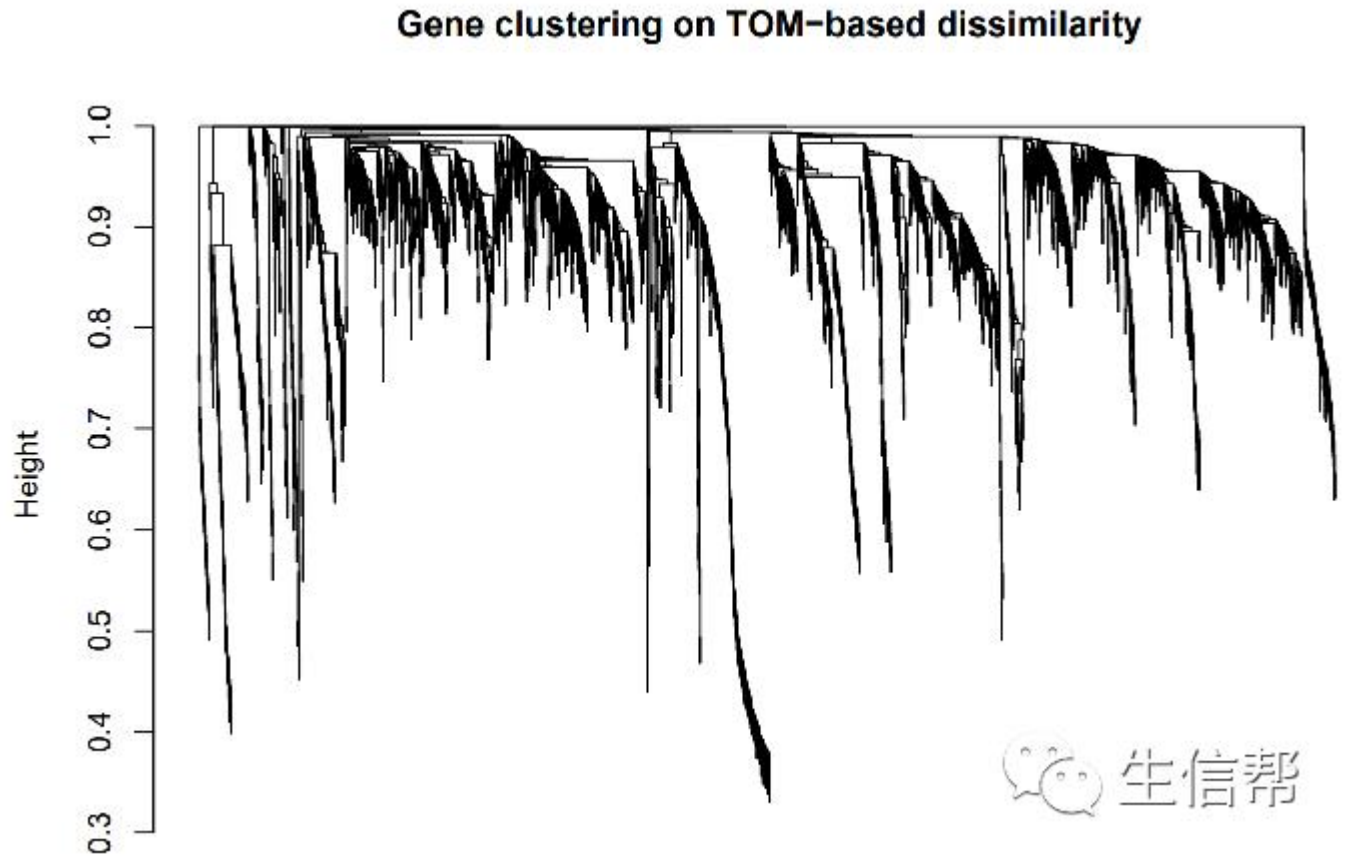
雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——计算基因间的相关系数

通过基因之间的相关系数转换成邻接矩阵（Adjacency Matrix），依据邻接矩阵构建TOM矩阵(topological overlap Matrix)，基于TOM矩阵构建分层聚类树，聚类树的不同分支代表不同的基因模块。

结果解读

TOM值表示基因间的相似度，将具有相似表达趋势的基因聚到一类。这些具有相似表达趋势的基因可以认为这些基因在功能上是相似的。



雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

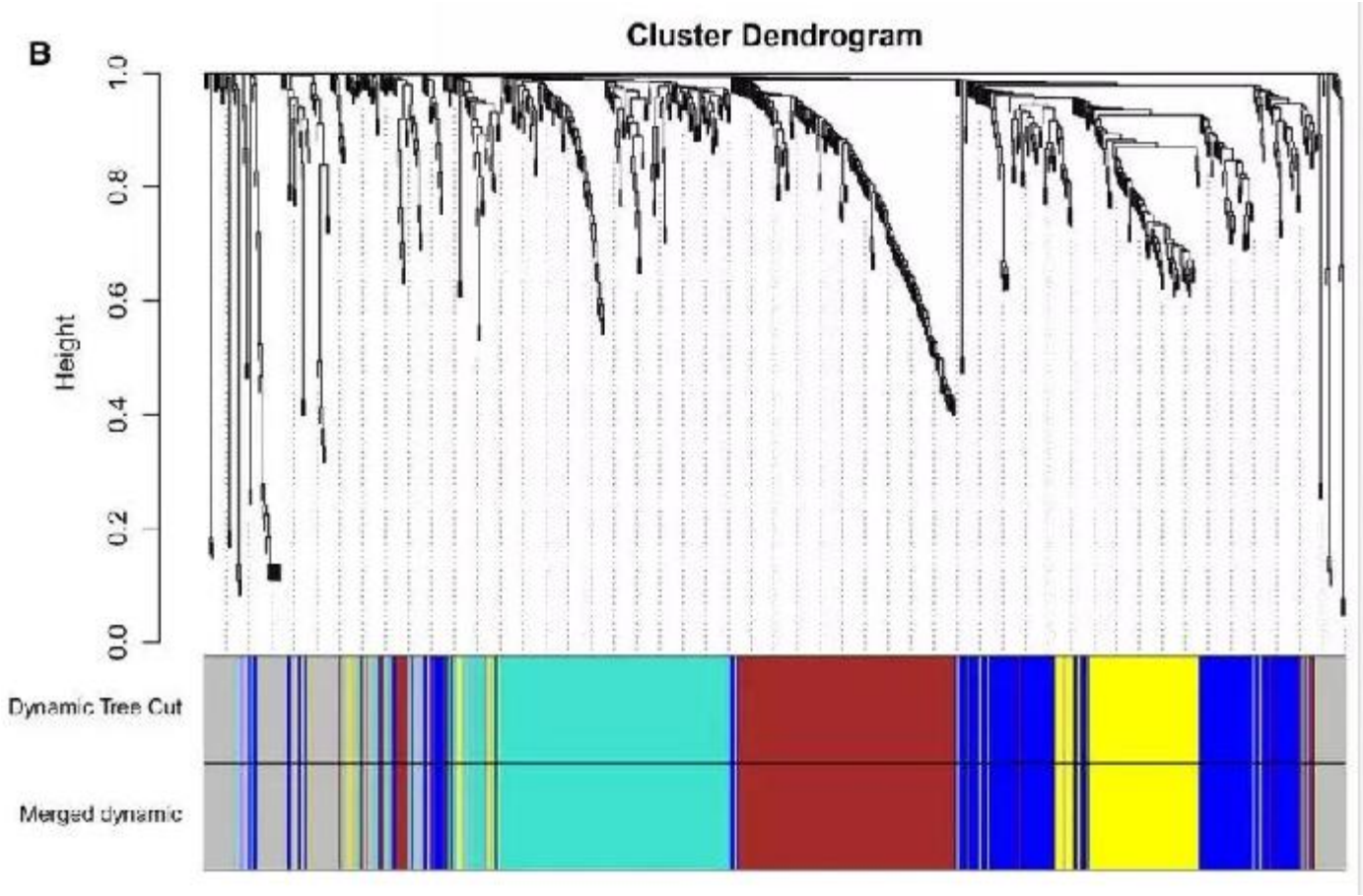
WGCNA主要步骤——鉴定基因模块

基于基因的加权相关系数计算出来的TOM值，将基因按照表达模式进行聚类，表达模式相似的基因归为一个模块（Module）。通常将模块树的分支采用dynamic branch cut methods，该方法的优点是可以自动化的将module区分开，不再需要手动指定高度。

结果解读

dynamic branch cut ，不同模块用不同颜色表示，同一模块的基因通常具有相似的功能。

Merged dynamic，要将相关性大于等于0.8的模块进行合并。



WGCNA主要步骤——基因模块与性状间的相关性

将上一步鉴定到的基因模块与外部性状进行关联分析，来看每个模块分别与哪个性状高度相关，从而找到我们感兴趣的性状对应的基因模块。

通过以下两个参数将每个模块与性状关联起来：

- 模块特征值与性状的相关系数；
- 模块与性状关联的显著性；

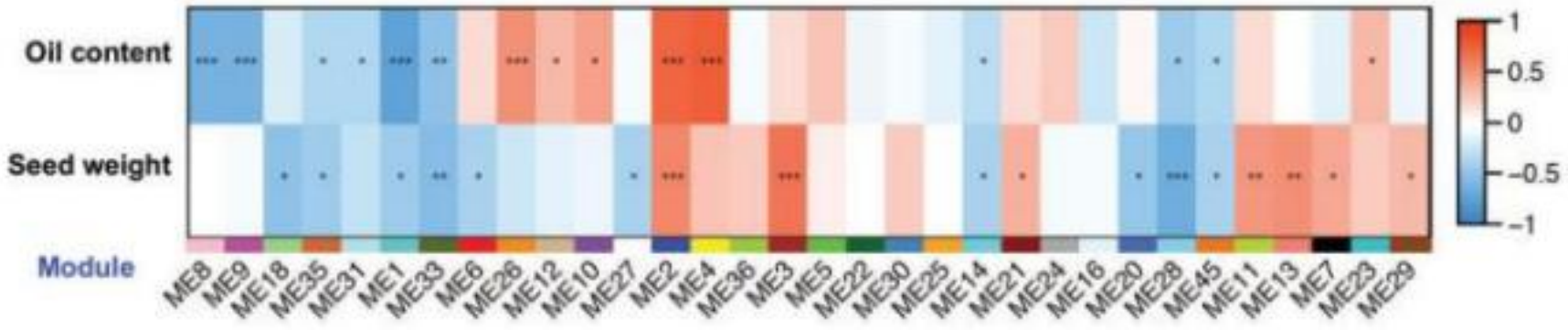
雲南農業大學
Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——基因模块与性状间的相关性

结果解读

WGCNA中使用PCA的方法计算模块矩阵的主成分，得到module eigenvalue（ME）。当ME越接近于0，则认为该性状与模块基因越不相关，而越接近于1或-1，则认为该性状与模块基因高度相关，正负表示其是正相关还是负相关。

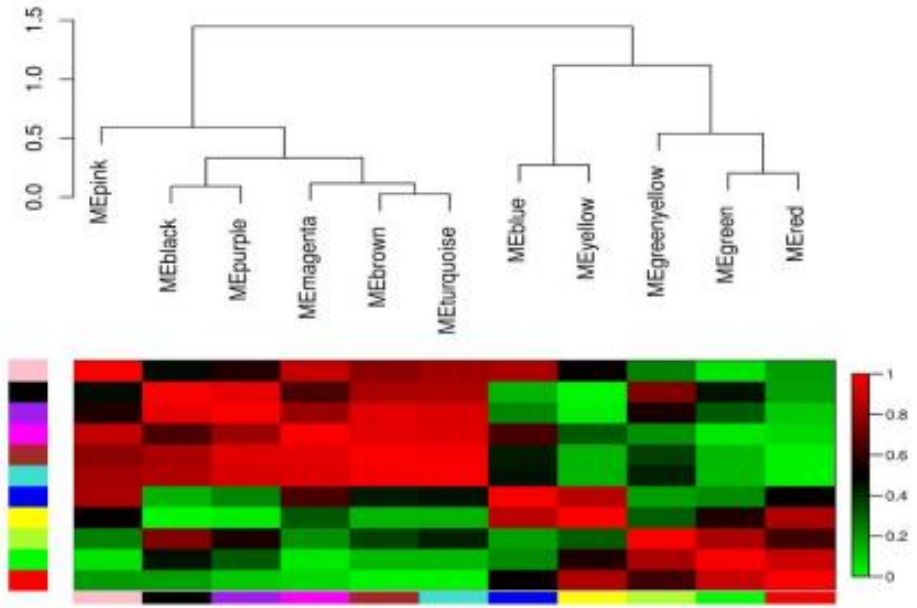
选择与某性状相关性最高的模块作为该性状的特征模块。



雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——模块间的相关性

模块之间不是孤立存在的，相互之间是有联系的，通过网络热图可以直观的看到性状关联模块与其它模块之间的联系强弱，从而鉴定各模块间的相关性。



结果解读

相关性越强的模块之间可能功能也相似；

在实际分析过程中，可以适当合并相关性强的模块做下一步分析。

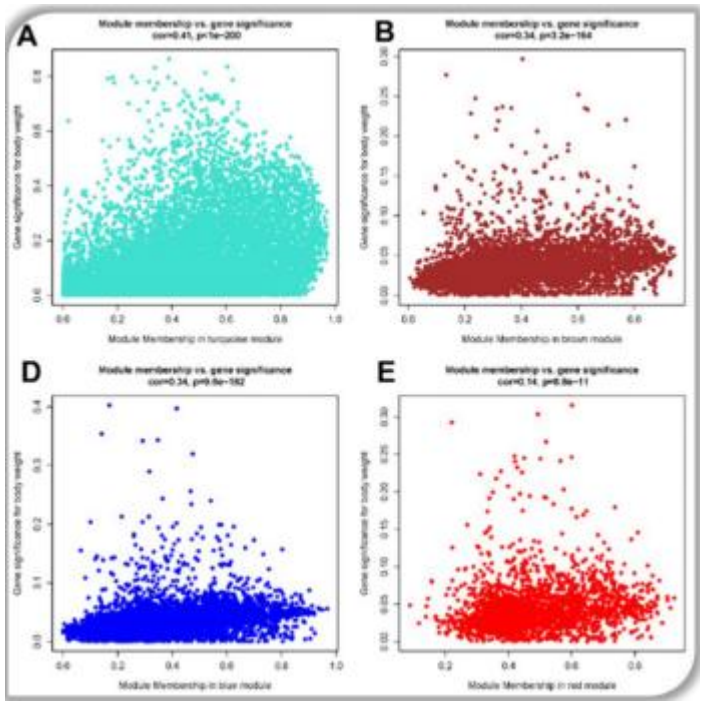
雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——计算MM值和GS值

用于鉴定模块中与性状最相关的基因。

module membership（MM）表示模块中的每个基因表达与模块间的相关性，MM值越接近于1或-1，则认为该基因与所在模块基因高度相关，正负表示其是正相关还是负相关

gene significance（GS）表示模块内基因与对应性状的显著性。



雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——计算MM值和GS值

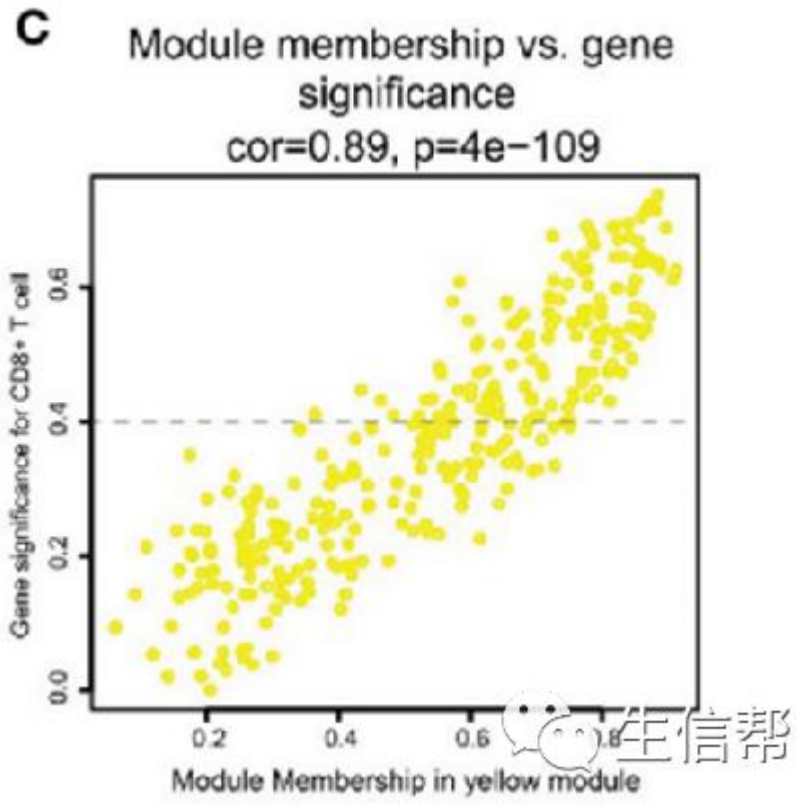
结果解读

右图横坐标表示基因与模块的相关性（MM），纵坐标表示基因与性状的显著性（GS）；

二者的相关性越强越好（斜对角分布）；

二者的相关性P值越低越显著；

如果MM和GS相关性较强，表明与性状高度相关的基因，通常也是该性状对应模块内比较重要的基因。因此后续在筛选核心基因做验证时应该考虑这两个值，优先推荐选取右上角部分的基因。



雲南農業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

WGCNA主要步骤——基因网络构建

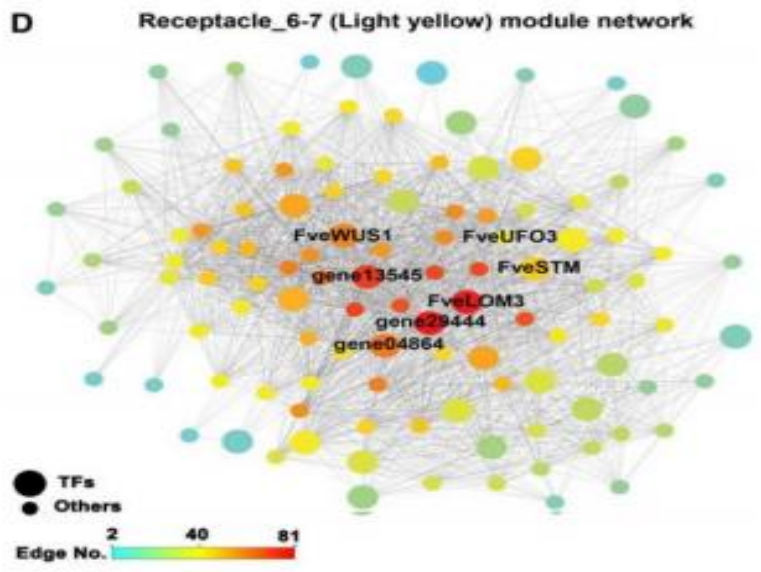
通过模块内部基因的关联网络，找到该模块的关键基因（hub gene），利用关系预测基因功能，并进行网络分析。

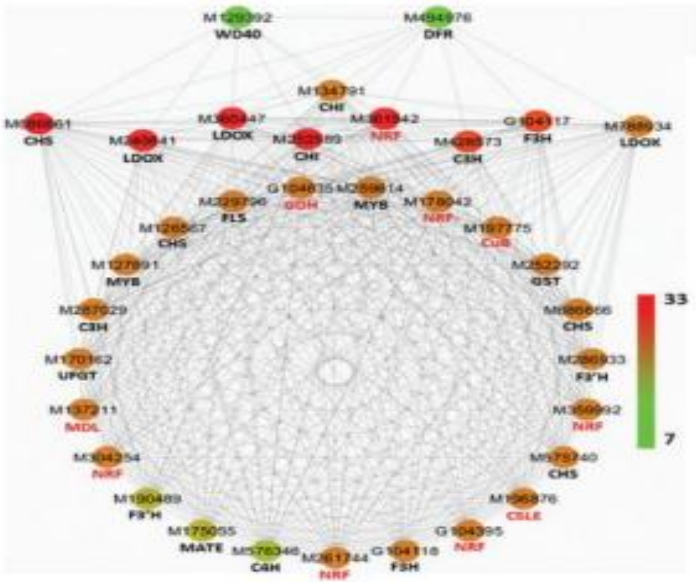
一般使用3个标准来筛选核心基因：

（1）基因与指定模块的显著性>0.2；

（2）基因与指定模块的相关性（MM）>0.8；

（3）q.weighted < 0.01





云南农業大學 Yunnan Agricultural University
云南省生物大数据重点实验室
生信帮

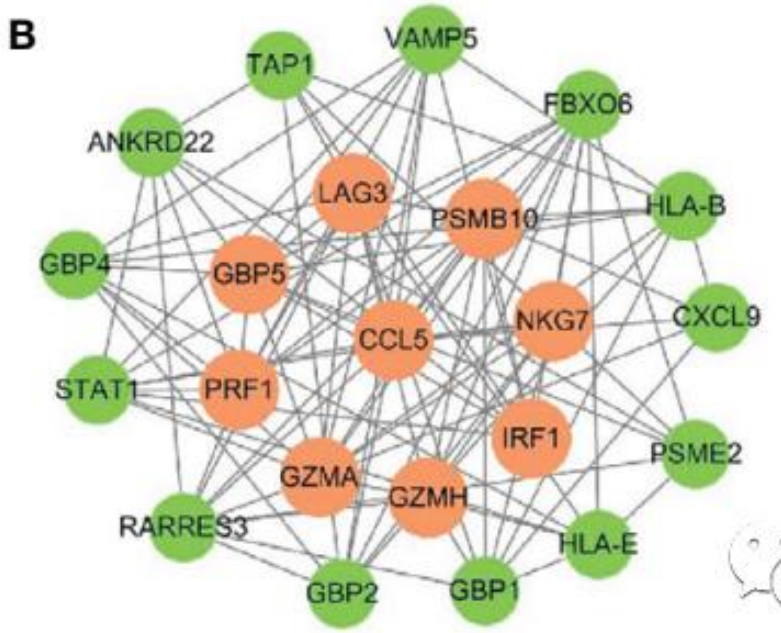
WGCNA主要步骤——基因网络构建

结果解读

越核心的基因与其他基因的连接度越多；

重点研究的基因以高亮的形式展现，突出体现；

不同类型基因可以用不同的形状展示；



生信帮，答你所问

雲南農業大學

Yunnan Agricultural University

云南省生物大数据重点实验室