

# 编程实战题目录

---

- 01: [生信编程很简单](#)
- 02: [人类基因组的外显子区域的长度](#)
- 03: [hg19基因组序列的一些探究](#)
- 04: [hg38每条染色体的基因、转录本分布](#)
- 05: [多个同样行列式文件的合并](#)
- 06: [根据GTF画基因的多个转录本结构](#)
- 07: [下载最新版的KEGG信息，并且解析好](#)
- 08: [写超几何分布检验](#)
- 09: [ID转换](#)
- 10: [根据指定染色体及坐标得到序列](#)
- 11: [根据指定染色体及坐标得到位置信息](#)
- 12: [把文件内容按照染色体分开写出](#)
- 13: [JSON格式数据的格式化](#)
- 14: [多个探针对应一个基因，取平均值、最大值](#)
- 15: [把counts矩阵转换成RPKM矩阵](#)
- 16: [对有临床信息的表达矩阵批量做生存分析](#)
- 17: [对多个差异分析结果直接取交集并集](#)
- 18: [根据GTF格式的基因注释文件得到人所有基因的染色体坐标](#)

## 00: 编程语言系统入门

---

- [生信分析人员如何系统入门python?](#)
- [生信分析人员如何系统入门perl?](#)
- [生信分析人员如何系统入门R?](#)
- [生信分析人员如何系统入门Linux?](#)

## 01: 生信编程很简单

---

### 题目

#### 对FASTQ的操作

- 5,3段截掉几个碱基
- 序列长度分布统计
- FASTQ 转换成 FASTA
- 统计碱基个数及GC%

#### 对FASTA的操作

- 取互补序列
- 取反向序列
- DNA to RNA
- 大小写字母形式输出
- 每行指定长度输出序列
- 按照序列长度/名字排序
- 提取指定ID的序列
- 随机抽取序列

#### 高级难度

- 根据坐标取序列
- 多文件合并
- 根据ID列表取序列
- GTF文件探索
- 简并碱基的引物序列还原成多条序列
- snp进行注释并格式化输出

#### 下载安装bowtie2（内含测试数据）

```
1 | cd ~/biosoft
2 | mkdir bowtie && cd bowtie
3 | wget https://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.9/bowtie2-2.2.9-linux-
  | x86_64.zip
4 | unzip bowtie2-2.2.9-linux-x86_64.zip
```

## 02: 人类基因组的外显子区域的长度

### 题目

下载人类外显子的坐标文件，编写代码统计外显子区域的长度。

### 测试数据

- Rbioconductor的TxDb.Hsapiens.UCSC.hg19.knownGene包
- NCBI数据库: [ftp://ftp.ncbi.nlm.nih.gov/pub/CCDS/current\\_human/](ftp://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_human/)

### R实现代码示例

```
1 rm(list=ls())
2 a=read.table(choose.files(),sep = '\t',stringsAsFactors = F,header = T) # 选择你下的CCDS文件
3 tmp <- apply(a[1:100,], 1, function(gene){ # 取前100行数据分析调试
4   # gene=a[3,]
5   x=gene[10] # Column10 外显子坐标位置列
6   if(grepl('\\]',x)){ # 判断x中是否存在有]这样的符号，如果有就利用正则替换掉。
7     x=sub('\\[', '', x)
8     x=sub('\\]', '', x)
9     # 这个时候得到的对象还是像这样的“880073-880179, 880436-880525.....”
10    tmp <- strsplit(as.character(x), ',')[[1]] # 我们先从逗号开始分割成小块
11    start <- as.numeric(unlist(lapply(tmp,function(y){ # 取开始位点
12      strsplit(as.character(y), '-')[[1]][1]
13    })))
14    end <- as.numeric(unlist(lapply(tmp,function(y){ # 取结束位点
15      strsplit(as.character(y), '-')[[1]][2]
16    })))
17    gene_d <- data.frame(gene=gene[3], # 将基因名，染色体，开始、结束位点绑定为数据框
18                        chr=gene[1],
19                        start=start,
20                        end=end
21    )
22    return (gene_d)#返回数据框
23  }
24 })
25 tmp_pos=c() # 构造一个空的向量
26 lapply(tmp[1:10], function(x){ # 取前10个list文件计算调试
27   # print(x)
28   if ( !is.null(x)){
29     apply(x, 1,function(y){
30       #print(y)
31       for ( i in as.numeric(y[3]):as.numeric(y[4]) ) # y[3]为坐标起点, y[4]为终止坐标, 历遍
32         tmp_pos <- c(tmp_pos,paste0(y[2], "-", i))
33     })
34   }
35 }
36 })
37 length(tmp_pos) # 计算exon的长度
38 length(unique(tmp_pos)) # 计算去重后的exon的长度
```



## 03: [hg19基因组序列的一些探究](#)

### 题目

求：hg19 每条染色体长度，每条染色体N的含量，GC含量。（高级作业：蛋白编码区域的GC含量会比基因组其它区域的高吗？）

### 测试数据

- hg19基因组序列下载

```
1 wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz # 也可以在浏览器上下
  载
2 tar xvzf chromFa.tar.gz
3 cat *.fa > hg19.fa
4 rm chr*.fa # 先把着急删，我待会可能要那他测试运行速度
```

- 简单测试数据

```
1 >chr_1
2 ATCGTCGaaAATGAANccNNttGTA
3 AGGTCTNAAccAAttGggG
4 >chr_2
5 ATCGAATGATCGANNGccTA
6 AGGTCTNAAAAGG
7 >chr_3
8 ATCGTCGANNGTAATggGA
9 AGGTCTNAAAAGG
10 >chr_4
11 ATCGTCaaaGANNAATGANGgggTA
```

### Perl代码示例

单行命令

```
1 perl -alne '{if(/^>/){chr=$_}else{ $A_count{chr}+=($_~tr/Aa//); $T_count{chr}+=
  ($_~tr/Tt//);$C_count{chr}+=($_~tr/Cc//); $G_count{chr}+=($_~tr/Gg//); $N_count{chr}+=
  ($_~tr/Nn//); }}END{print
  "$_\t$A_count{$_}\t$T_count{$_}\t$C_count{$_}\t$G_count{$_}\t$N_count{$_}" foreach sort keys
  %N_count}' test.fa
```

完整代码

```

1  while (<>){
2  chomp;
3  if(/^>/){
4  $chr=$_
5  }
6  else{
7  $A_count{$chr}+=$( $_=~tr/Aa//);
8  $T_count{$chr}+=$( $_=~tr/Tt//);
9  $C_count{$chr}+=$( $_=~tr/Cc//);
10 $G_count{$chr}+=$( $_=~tr/Gg//);
11 $N_count{$chr}+=$( $_=~tr/Nn//);
12 }
13 }
14 foreach (sort keys %N_count){
15 $length = $A_count{$_}+$T_count{$_}+$C_count{$_}+$G_count{$_}+$N_count{$_};
16 $gc = ($G_count{$_}+$C_count{$_})/($A_count{$_}+$T_count{$_}+$C_count{$_}+$G_count{$_});
17 print
18 "$_\\t$A_count{$_}\\t$T_count{$_}\\t$C_count{$_}\\t$G_count{$_}\\t$N_count{$_}\\t$length\\t$gc\\n"
19 }

```

## 参考结果

结果如下：

|   |        |    |    |   |    |   |
|---|--------|----|----|---|----|---|
| 1 | >chr_1 | 13 | 10 | 7 | 10 | 4 |
| 2 | >chr_2 | 11 | 6  | 5 | 8  | 4 |
| 3 | >chr_3 | 10 | 6  | 3 | 10 | 4 |
| 4 | >chr_4 | 9  | 4  | 2 | 7  | 3 |

## 04: [hg38每条染色体的基因、转录本分布](#)

### 题目

对GTF注释文件进行探究，统计每条染色体基因数、转录本数、内含子数、外显子数。（高级作业：下载 human/rat/mouse/dog/cat/chicken等物种的gtf注释文件

（<http://asia.ensembl.org/info/data/ftp/index.html>），编写函数实现对多个GTF文件进行批量统计染色体基因、转录本的分布及转录本外显子个数；继续探索回答以下问题：所有基因平均有多少个转录本？所有转录本平均有多少个exon和intron？如果要比较多个数据库呢（gencode/UCSC/NCBI？）？如果把基因分成多个类型呢？protein coding gene, pseudogene, lncRNA还有miRNA的基因？它们的特征又有哪些变化呢？）

### 测试数据

```
1 wget -c ftp://ftp.ensembl.org/pub/release-
  87/gtf/homo_sapiens/Homo_sapiens.GRCh38.87.chr.gtf.gz
2 gzip -d Homo_sapiens.GRCh38.87.chr.gtf.gz
```

### 代码示例

```
1 # 每条染色体的基因个数
2 zcat Homo_sapiens.GRCh38.87.chr.gtf.gz | perl -alne '{print if $F[2] eq "gene" }' | cut -f 1
  | sort | uniq -c
3 # 基因分类
4 zcat Homo_sapiens.GRCh38.87.chr.gtf.gz | perl -alne '{next unless $F[2] eq "gene"
  ;/gene_biotype "(.*?)"/;print $1}' | sort | uniq -c
```



# 05: 多个同样行列式文件的合并

## 题目

将htseq-count生成的所有独立样本文件进行合并（每个样品对应一个文件，包括了所有基因表达量）。希望通过编程处理每个文件得到输出的表达矩阵（行名是基因名，列名是样品名），如下所示：

|                  | 184A1        | 184B5        | 21MT1        | 21MT2        | 21NT         | 21PT         |
|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| ENSG00000000003  | 9.521255e+01 | 9.569868e+01 | 1.999467e+01 | 6.568638e+01 | 4.405775e+01 | 3.431757e+01 |
| ENSG00000000005  | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 1.492021e-01 | 0.000000e+00 | 0.000000e+00 |
| ENSG000000000419 | 4.532083e+02 | 2.436480e+02 | 1.420582e+02 | 2.004131e+02 | 1.931544e+02 | 1.515729e+02 |
| ENSG000000000457 | 1.810439e+01 | 2.656661e+01 | 1.612776e+01 | 1.208731e+01 | 1.844801e+01 | 1.578353e+01 |
| ENSG000000000460 | 4.816622e+01 | 2.458429e+01 | 2.428459e+01 | 3.651692e+01 | 3.258676e+01 | 2.852255e+01 |
| ENSG000000000938 | 3.060651e+00 | 3.158946e-01 | 3.145795e-01 | 0.000000e+00 | 1.328697e-01 | 0.000000e+00 |
| ENSG000000000971 | 2.044634e+01 | 2.104032e+00 | 2.340722e+00 | 3.579494e+00 | 2.723789e+01 | 2.249986e+01 |
| ENSG000000001036 | 1.558110e+02 | 2.859372e+02 | 2.446551e+02 | 3.635973e+02 | 3.534835e+02 | 3.473198e+02 |
| ENSG000000001084 | 9.813004e+01 | 2.612476e+02 | 1.017421e+02 | 2.103850e+02 | 1.811553e+02 | 1.577041e+02 |
| ENSG000000001167 | 8.951262e+01 | 9.415437e+01 | 2.454335e+02 | 2.164871e+02 | 2.860924e+02 | 2.892983e+02 |
| ENSG000000001460 | 2.965779e+01 | 2.506156e+01 | 2.044668e+01 | 1.496007e+01 | 1.467831e+01 | 1.652478e+01 |
| ENSG000000001461 | 1.742302e+02 | 2.785649e+02 | 5.810591e+01 | 5.924132e+01 | 7.616939e+01 | 7.157401e+01 |

## 测试数据

### 模拟数据

用perl脚本模仿htseq-count计算每个样本所有的基因表达量的输出独立样本文件：

```
1 ## 首先新建文件tmp.sh 输入这个代码：
2 perl -le '{print "gene_$_\t".int(rand(1000)) foreach 1..99}'
3 ## 然后用perl脚本调用这个tmp.sh文件：
4 perl -e 'system(" bash tmp.sh >$_ .txt") foreach a..z'
5 ## 这样就生成了a~z这26个样本的counts文件
```

第一列是基因，第二列是该基因的counts值，共有a~z这26个样本的counts文件，需要合并成一个大的行列式，这样才能导入到R里面做差异分析，如果手工用excel表格做，当然是可以的，但是太麻烦，如果有500个样本，正常人都不会去手工做了，需要编程。

每个样本的基因顺序并不一致，这时候你应该怎么做呢？

### 真实数据

实际需求如下：[GSE48213](#)里面有56个文件，需要合并成一个表达矩阵，来根据cell-line的不同，分组做差异分析。可以查看[paper](#)

```
1 wget -c ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE48nnn/GSE48213/suppl/GSE48213_RAW.tar
2 tar -xf GSE48213_RAW.tar
3 gzip -d *.gz
```

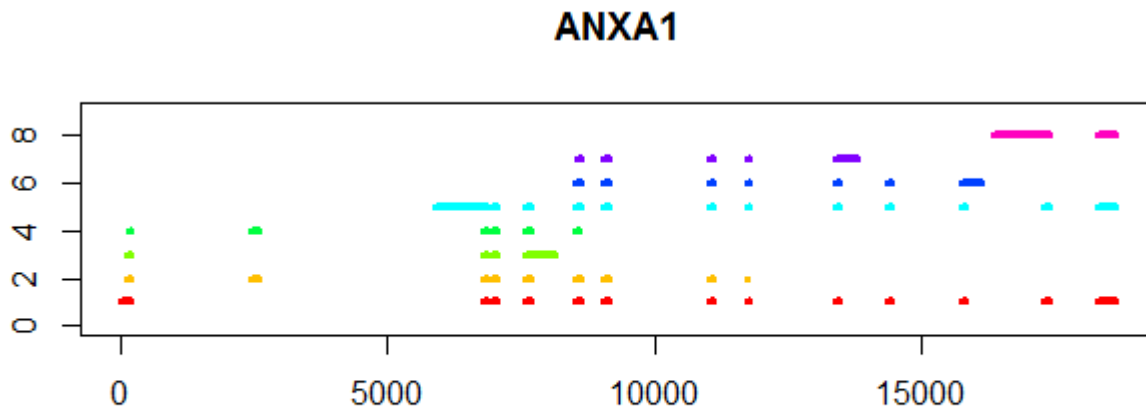
## 代码示例

```
1 ## 首先在GSE48213_RAW目录里面生成tmp.txt文件，使用shell脚本：
2 awk '{print FILENAME"\t"$0}' * |grep -v Ensembl_Gene_ID >tmp.txt
3 ## 然后把tmp.txt导入R语言里面用reshape2处理即可！
4 setwd('tmp/GSE48213_RAW/')
5 a=read.table('tmp.txt',sep = '\t',stringsAsFactors = F)
6 library(reshape2)
7 fpkm <- dcast(a,formula = V2~V1)
```

## 06: 根据GTF画基因的多个转录本结构

### 题目

从NCBI,ENSEMBL,UCSC,GENCODE数据库下载各种GTF注释文件，编写代码得到所有基因的转录本个数，以及每个转录本的外显子的坐标，绘制如下转录本结构图：



**chr9:75766673-75785309**

比如对这个ANXA1基因来说，非常多的转录本，但是基因的起始终止坐标，是所有转录本起始终止坐标的极大值和极小值。同时，它是一个闭合基因，因为它存在一个转录本的起始终止坐标等于该基因的起始终止坐标。可以看到它的外显子并不是非常整齐的，虽然多个转录本会共享某些外显子，但是也存在某些外显子比同区域其它外显子长的现象。

### 测试数据

```
1 wget -c  
  http://www.broadinstitute.org/cancer/cga/sites/default/files/data/tools/rnaseqc/gencode.v7.anno-  
  notation_goodContig.gtf.gz  
2 gzip -d gencode.v7.annotation_goodContig.gtf.gz
```

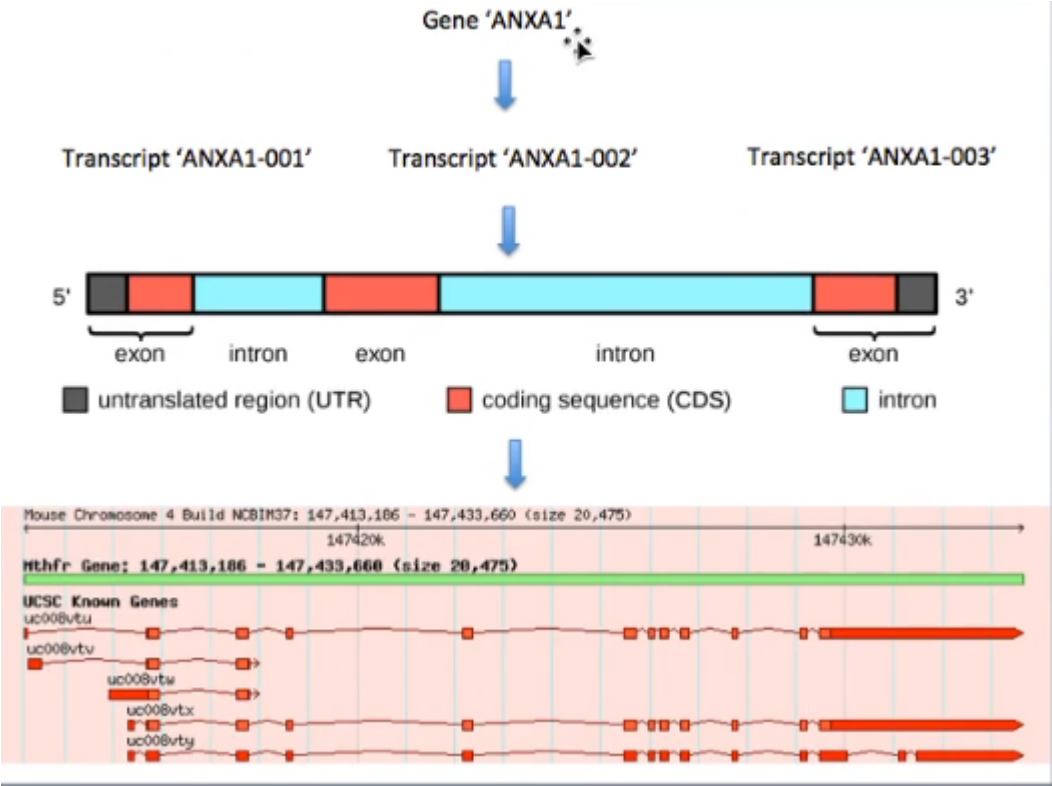
### R实现代码示例

```

1 rm(list=ls())
2
3 ##
4 http://www.broadinstitute.org/cancer/cga/sites/default/files/data/tools/rnaseqc/gencode.v7.annotation\_goodContig.gtf.gz
5 setwd('tmp')
6 gtf <- read.table('gencode.v7.annotation_goodContig.gtf.gz',stringsAsFactors = F,
7                 header = F,comment.char = "#",sep = '\t'
8                 )
9 table(gtf[,2])
10 gtf <- gtf[gtf[,2] == 'HAVANA',]
11 gtf <- gtf[grepl('protein_coding',gtf[,9]),]
12
13 lapply(gtf[1:10,9], function(x){
14   y=strsplit(x,';')
15 })
16
17 gtf$gene <- lapply(gtf[,9], function(x){
18   y <- strsplit(x,';')[[1]][5]
19   strsplit(y,'\\s')[[1]][3]
20   }
21 )
22 draw_gene = 'TP53'
23 structure = gtf[gtf$gene==draw_gene,]
24
25 colnames(structure) =c(
26   'chr','db','record','start','end','tmp1','tmp2','tmp3','tmp4','gene'
27 )
28 gene_start <- min(c(structure$start,structure$end))
29 gene_end <- max(c(structure$start,structure$end))
30 tmp_min=min(c(structure$start,structure$end))
31 structure$new_start=structure$start-tmp_min
32 structure$new_end=structure$end-tmp_min
33 tmp_max=max(c(structure$new_start,structure$new_end))
34 num_transcripts=nrow(structure[structure$record=='transcript',])
35 tmp_color=rainbow(num_transcripts)
36
37 x=1:tmp_max;y=rep(num_transcripts,length(x))
38 #x=10000:17000;y=rep(num_transcripts,length(x))
39 plot(x,y,type = 'n',xlab='',ylab = '',ylim = c(0,num_transcripts+1))
40 title(main = draw_gene,sub = paste("chr",structure$chr,":",gene_start,"-",gene_end,sep=""))
41 j=0;
42 tmp_legend=c()
43 for (i in 1:nrow(structure)){
44   tmp=structure[i,]
45   if(tmp$record == 'transcript'){
46     j=j+1
47     tmp_legend=c(tmp_legend,paste("chr",tmp$chr,":",tmp$start,"-",tmp$end,sep=""))
48   }
49   if(tmp$record == 'exon') lines(c(tmp$new_start,tmp$new_end),c(j,j),col=tmp_color[j],lwd=4)
50 }

```

参考结果



## 07: [下载最新版的KEGG信息，并且解析好](#)

### 题目

下载最新版的KEGG注释文本文件，编写脚本整理成kegg的pathway的ID与基因ID的对应格式。

### 测试数据

- 1 首先打开[KEGG官方网站](#)，网页中展示出了各个物种的分类、拉丁名称、英文名称等信息。



#### KEGG Organisms: Complete Genomes

Eukaryotes: 360 Bacteria: 4058 Archaea: 244

[ Genomes | Species | Genus | Viruses | Meta ]

#### Eukaryotes

| Category |  |      | Organisms  | Source |
|----------|--|------|--|--------|
|          |  | hsa  | Homo sapiens (human)                                       | RefSeq |
|          |  | ptr  | Pan troglodytes (chimpanzee)                               | RefSeq |
|          |  | pps  | Pan paniscus (bonobo)                                      | RefSeq |
|          |  | ggo  | Gorilla gorilla gorilla (western lowland gorilla)          | RefSeq |
|          |  | pon  | Pongo abelii (Sumatran orangutan)                          | RefSeq |
|          |  | nle  | Nomascus leucogenys (northern white-cheeked gibbon)        | RefSeq |
|          |  | mcc  | Macaca mulatta (rhesus monkey)                             | RefSeq |
|          |  | mcf  | Macaca fascicularis (crab-eating macaque)                  | RefSeq |
|          |  | csab | Chlorocebus sabaeus (green monkey)                         | RefSeq |
|          |  | rro  | Rhinopithecus roxellana (golden snub-nosed monkey)         | RefSeq |
|          |  | rbb  | Rhinopithecus bieti (black snub-nosed monkey)              | RefSeq |
|          |  | cjc  | Callithrix jacchus (white-tufted-ear marmoset)             | RefSeq |
|          |  | sbq  | Saimiri boliviensis boliviensis (Bolivian squirrel monkey) | RefSeq |
|          |  | mmu  | Mus musculus (mouse)                                       | RefSeq |
|          |  | rno  | Rattus norvegicus (rat)                                    | RefSeq |
|          |  | cge  | Cricetulus griseus (Chinese hamster)                       | RefSeq |

- 2 直接网页中搜索（Ctrl + F）需要下载的物种英文名称或拉丁名。如果不确定物种名称，网站中提供了详细的分类系统，也可根据前面的物种分类信息进行查找。

本文以拟南芥为例，搜索“*Arabidopsis thaliana*”即可找到。找到后点击物种名称前的3个字母缩写链接（下图红色框中的位置）。

|                |     |                                       |        |
|----------------|-----|---------------------------------------|--------|
| Cnidarians     | adt | Acropora digitata (stony coral)       | RefSeq |
|                | hmg | Hydra vulgaris                        | RefSeq |
|                | tad | Trichoplax adhaerens                  | RefSeq |
|                | aqu | Amphimedon queenslandica (sponge)     | RefSeq |
| Placozoans     | ath | Arabidopsis thaliana (thale cress)    | RefSeq |
| Poriferans     | aly | Arabidopsis lyrata (lyrate rockcress) | RefSeq |
|                | crb | Capsella rubella                      | RefSeq |
|                | eus | Eutrema salsugineum                   | RefSeq |
|                | bro | Brassica rapa (field mustard)         | RefSeq |
| Mustard family |     |                                       |        |

- 3 进入后的网页中包含了物种的一些基因组信息，点击上方的“Brite hierarchy”，进入后再点击“KEGG Orthology (KO)”；

## KEGG Orthology (KO) - Homo sapiens (human)

[ [Brite menu](#) | [Organism menu](#) | [Download htext](#) ]

Homo sapiens (human)

Go

▼ ▼ ▼ ▼ ☐ One-click mode

KO

- ▶ Metabolism
- ▶ Genetic Information Processing
- ▶ Environmental Information Processing
- ▶ Cellular Processes
- ▶ Organismal Systems
- ▶ Human Diseases

[ [BRITE](#) | [KEGG2](#) | [KEGG](#) ]

Last updated: February 17, 2017

- 4 在跳转出的网页中点击“Download htext”，弹出下载窗口进行下载，国外网站有时会出现无法下载的情况，多试几次即可；

## KEGG Homo sapiens (human)

| Genome info   | Pathway map | Brite hierarchy | Module | Genome map | Blast | Taxonomy |
|---|-------------|-----------------|--------|------------|-------|----------|
| Search genes: <input type="text"/> <input type="button" value="Go"/> <input type="button" value="Clear"/> |             |                 |        |            |       |          |
| KEGG Mapper: <a href="#">Search Brite</a><br><a href="#">Search&amp;Color Brite</a>                       |             |                 |        |            |       |          |
| <b>BRITE functional hierarchies</b>   |             |                 |        |            |       |          |
| 00001 KEGG Orthology (KO) ←   |             |                 |        |            |       |          |
| 01000 Enzymes   |             |                 |        |            |       |          |
| 01001 Protein kinases   |             |                 |        |            |       |          |
| 01009 Protein phosphatases and associated proteins  |             |                 |        |            |       |          |
| 01002 Peptidases  |             |                 |        |            |       |          |
| 01003 Glycosyltransferases  |             |                 |        |            |       |          |
| 01005 Lipopolysaccharide biosynthesis proteins  |             |                 |        |            |       |          |
| 01004 Lipid biosynthesis proteins   |             |                 |        |            |       |          |
| 01008 Polyketide biosynthesis proteins  |             |                 |        |            |       |          |
| 01006 Prenyltransferases  |             |                 |        |            |       |          |
| 01007 Amino acid related enzymes  |             |                 |        |            |       |          |
| 00199 Cytochrome P450   |             |                 |        |            |       |          |
| 00194 Photosynthesis proteins   |             |                 |        |            |       |          |
| 03000 Transcription factors   |             |                 |        |            |       |          |
| 03021 Transcription machinery   |             |                 |        |            |       |          |
| 03019 Messenger RNA biogenesis  |             |                 |        |            |       |          |

- 5 当然，下载好之后还没有结束。下载得到文本文件，可以看到里面的结构层次非常清楚，C开头的就是kegg的pathway的ID所在行，D开头的就是属于它的kegg的所有的基因。A,B是kegg的分类，总共是6个大类，42

个小类。

```
1 +D GENES KO
2 #<h2><a href="/kegg/kegg2.html"></a> &nbsp; KEGG Orthology (KO) - Homo sapiens (human) <
3 %<style type="text/css"><!-- #grid{ table-layout:fixed; font-family: monospace; position: relative; width: 1200px; color: black; }.col1{ positi
4 #<!--
5 #ENTRY hsa00001
6 #NAME T01001
7 #DEFINITION KEGG Orthology (KO) - Homo sapiens (human)
8 #<!--
9 !
10 A<b>Metabolism</b>
11 B
12 B <b>Overview</b>
13 C 01200 Carbon metabolism [PATH:hsa01200]
14 D 3101 HK3; hexokinase 3 K00844 HK; hexokinase [EC:2.7.1.1]
15 D 3098 HK1; hexokinase 1 K00844 HK; hexokinase [EC:2.7.1.1]
16 D 3099 HK2; hexokinase 2 K00844 HK; hexokinase [EC:2.7.1.1]
17 D 80201 HKDC1; hexokinase domain containing 1 K00844 HK; hexokinase [EC:2.7.1.1]
18 D 2645 GCK; glucokinase K12407 GCK; glucokinase [EC:2.7.1.2]
19 D 83440 ADPGK; ADP dependent glucokinase K08074 ADPGK; ADP-dependent glucokinase [EC:2.7.1.147]
20 D 2821 GPI; glucose-6-phosphate isomerase K01810 GPI; glucose-6-phosphate isomerase [EC:5.3.1.9]
21 D 5213 PFKM; phosphofructokinase, muscle K00850 pfkA; 6-phosphofructokinase 1 [EC:2.7.1.11]
22 D 5214 PFKP; phosphofructokinase, platelet K00850 pfkA; 6-phosphofructokinase 1 [EC:2.7.1.11]
23 D 5211 PFKL; phosphofructokinase, liver type K00850 pfkA; 6-phosphofructokinase 1 [EC:2.7.1.11]
24 D 230 ALDOC; aldolase, fructose-bisphosphate C K01623 ALDO; fructose-bisphosphate aldolase, class I [EC:4.1.2.13]
25 D 226 ALDOA; aldolase, fructose-bisphosphate A K01623 ALDO; fructose-bisphosphate aldolase, class I [EC:4.1.2.13]
26 D 229 ALDOB; aldolase, fructose-bisphosphate B K01623 ALDO; fructose-bisphosphate aldolase, class I [EC:4.1.2.13]
27 D 7167 TPI1; triosephosphate isomerase 1 K01803 TPI; triosephosphate isomerase (TIM) [EC:5.3.1.1]
28 D 2597 GAPDH; glyceraldehyde-3-phosphate dehydrogenase K00134 GAPDH; glyceraldehyde 3-phosphate dehydrogenase [EC:1.2.1.12]
29 D 5232 PGK2; phosphoglycerate kinase 2 K00927 PGK; phosphoglycerate kinase [EC:2.7.2.3]
```

## Perl代码示例

```
1 perl -alne 'if(/^C/){/PATH:hsa(\d+)/;$kegg=$1}else{print "$kegg\t$F[1]" if /^D/ and $kegg;}}' hsa00001.keg >kegg2gene.txt
```

## 参考结果

```
jmzeng@ubuntu:/home/jmzeng/tmp/kegg$ head kegg2gene.txt
01200 3101
01200 3098
01200 3099
01200 80201
01200 2645
01200 83440
01200 2821
01200 5213
01200 5214
01200 5211
jmzeng@ubuntu:/home/jmzeng/tmp/kegg$ cut -f 1 kegg2gene.txt |sort -u |wc
299 299 1794
jmzeng@ubuntu:/home/jmzeng/tmp/kegg$ cut -f 2 kegg2gene.txt |sort -u |wc
6992 6992 38421
jmzeng@ubuntu:/home/jmzeng/tmp/kegg$
```



## 08: 写超几何分布检验

---

### 题目

学习GO/KEGG的富集分析的原理，编写代码实现超几何分布检验，将得到的结果与测试数据中的kegg.enrichment.html进行比较。

(P值的计算:  $C(k,M)*C(n-k,N-M)/C(n,N)$  )

### 测试数据

- kegg2gene（第六讲kegg数据解析结果）


暂时不用最新版的kegg注释数据，为了能够统一答案





- 差异基因list和背景基因list(R代码)

```

1 source("http://bioconductor.org/biocLite.R")
2 biocLite("org.Hs.eg.db")
3 biocLite("KEGG.db")
4 biocLite("GOstats")
5 biocLite("hgu95av2.db")
6
7 library(org.Hs.eg.db)
8 library(KEGG.db)
9 library(GOstats)
10 library("hgu95av2.db")
11
12 ##得到kegg2gene.list(KEGG注释信息)
13
14 tmp=toTable(org.Hs.egPATH)
15 write.table(tmp, 'kegg2gene.list.txt',quote = F,row.names = F)
16 ## 得到universeGeneIds.txt(背景基因list)
17 ls('package:hgu95av2.db')
18 universeGeneIds <- unique(mappedRkeys(hgu95av2ENTREZID))
19 write.table(universeGeneIds, 'universeGeneIds.txt',quote = F,row.names = F)
20
21 ## 得到diff_gene.txt(差异基因list)
22
23 set.seed('123456.789')
24 diff_gene = sample(universeGeneIds,300)
25 write.table(diff_gene, 'diff_gene.txt',quote = F,row.names = F)
26
27 ## 得到kegg.enrichment.html(富集分析结果)
28
29 annotationPKG='org.Hs.eg.db'
30 hyperG.params = new("KEGGHyperGParams", geneIds=diff_gene, universeGeneIds=universeGeneIds,
31 annotation=annotationPKG,
32 categoryName="KEGG", pvalueCutoff=1, testDirection = "over")
33 KEGG.hyperG.results = hyperGTest(hyperG.params);
34 htmlReport(KEGG.hyperG.results, file="kegg.enrichment.html",
35 summary.args=list("htmlLinks"=TRUE))

```

 > This PC > Local Disk (D:) > 生信技能树-视频直播 > 第七讲 > data

| <input type="checkbox"/> Name  |          | Date modified    | Type              | Size   |
|--|----------|------------------|-------------------|--------|
|  diff_gene.txt        | ← 差异基因   | 23/02/2017 20:54 | Text Document     | 2 KB   |
|  kegg.enrichment.html | ← 富集分析结果 | 23/02/2017 20:54 | Chrome HTML Do... | 38 KB  |
|  kegg2gene.list.txt   | ← KEGG数据 | 23/02/2017 20:53 | Text Document     | 195 KB |
|  universeGeneIds.txt  | ← 背景基因   | 23/02/2017 20:54 | Text Document     | 53 KB  |

## R代码示例

下载链接: [https://github.com/jmzeng1314/humanid/blob/master/R/hyperGtest\\_jimmy.R](https://github.com/jmzeng1314/humanid/blob/master/R/hyperGtest_jimmy.R)

```

1 library("hgu95av2.db")[/align][align=left]ls('package:hgu95av2.db')
2 universeGeneIds <- unique(mappedRkeys(hgu95av2ENTREZID))
3 set.seed('123456.789')
4 diff_gene = sample(universeGeneIds,300)
5 library(org.Hs.eg.db)
6 library(KEGG.db)
7 tmp=toTable(org.Hs.egPATH)
8 GeneID2kegg<- tapply(tmp[,2],as.factor(tmp[,1]),function(x) x)
9 kegg2GeneID<- tapply(tmp[,1],as.factor(tmp[,2]),function(x) x)
10 #results <- hyperGtest_jimmy(GeneID2kegg,kegg2GeneID,diff_gene,universeGeneIds)
11
12 GeneID2Path=GeneID2kegg
13 Path2GeneID=kegg2GeneID
14
15 diff_gene_has_path=intersect(diff_gene,names(GeneID2Path))
16 n=length(diff_gene) #306
17 N=length(universeGeneIds) #5870
18 results=c()
19
20 for (i in names(Path2GeneID)){
21   #i="04672"
22   M=length(intersect(Path2GeneID[[i]],universeGeneIds))
23   #print(M)
24
25   if(M<5)
26     next
27   exp_count=n*M/N
28   #print(paste(n,N,M,sep="\t"))
29   k=0
30   for (j in diff_gene_has_path){
31     if (i %in% GeneID2Path[[j]]) k=k+1
32   }
33   OddsRatio=k/exp_count
34   if (k==0) next
35   p=phyper(k-1,M, N-M, n, lower.tail=F)
36   #print(paste(i,p,OddsRatio,exp_count,k,M,sep=" "))
37   results=rbind(results,c(i,p,OddsRatio,exp_count,k,M))
38 }
39 colnames(results)=c("PathwayID", "Pvalue", "OddsRatio", "ExpCount", "Count", "Size")
40 results=as.data.frame(results,stringsAsFactors = F)
41 results$p.adjust = p.adjust(results$Pvalue,method = 'BH')
42 results=results[order(results$Pvalue),]
43 rownames(results)=1:nrow(results)

```

## 09: ID转换

### 题目

probe\_id, gene\_id, gene\_name, symbol之间的转换。

### 测试数据

- 需要转换的探针ID(变量my\_probe)

```
1 rm(list=ls())
2 library("hgu95av2.db")
3 ls('package:hgu95av2.db')
4 probe2entrezID=toTable(hgu95av2ENTREZID)
5 probe2symbol=toTable(hgu95av2SYMBOL)
6 probe2genename=toTable(hgu95av2GENENAME)
7
8 my_probe = sample(unique(mappedLkeys(hgu95av2ENTREZID)),30)
9
10 tmp1 = probe2symbol[match(my_probe,probe2symbol$probe_id),]
11 tmp2 = probe2entrezID[match(my_probe,probe2entrezID$probe_id),]
12 tmp3 = probe2genename[match(my_probe,probe2genename$probe_id),]
13
14 write.table(my_probe,'my_probe.txt',quote = F,col.names = F,row.names =F)
15 write.table(tmp1$symbol,'my_symbol.txt',quote = F,col.names = F,row.names =F)
16 write.table(tmp2$gene_id,'my_geneID.txt',quote = F,col.names = F,row.names =F)
```

- 下载对应关系  
ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/
- illumina芯片的探针

所有bioconductor支持的芯片平台对应关系：通过bioconductor包来获取所有的芯片探针与gene的对应关系；可以从NCBI的GPL平台下载：<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL6947>；也可以直接加载对应的包；或者直接去公司的主页下载manifest文件。

```
1 library("illuminaHumanv4.db")
2 ls('package:illuminaHumanv4.db')
3
4 probe2entrezID=toTable(illuminaHumanv4ENTREZID)
5 probe2symbol=toTable(illuminaHumanv4SYMBOL)
6 probe2genename=toTable(illuminaHumanv4GENENAME)
7
8 my_probe = sample(unique(mappedLkeys(illuminaHumanv4ENTREZID)),30)
9
10 probe2symbol[match(my_probe,probe2symbol$probe_id),]
11 probe2entrezID[match(my_probe,probe2entrezID$probe_id),]
12 probe2genename[match(my_probe,probe2genename$probe_id),]
13
```

### R代码示例

基因的转换：运行下面的R代码，得到的my\_symbol\_gene和my\_entrez\_gene就是需要转换的ID。

```
1 library("illuminaHumanv4.db")
2 ls('package:illuminaHumanv4.db')
3 my_entrez_gene = sample(unique(mappedRkeys(illuminaHumanv4ENTREZID)),30)
4 my_symbol_gene = sample(unique(mappedRkeys(illuminaHumanv4SYMBOL)),30)
5
6 library("org.Hs.eg.db")
7 ls('package:org.Hs.eg.db')
8
9 entrezID2symbol <- toTable(org.Hs.egSYMBOL)
10
11 entrezID2symbol[match(my_entrez_gene,entrezID2symbol$gene_id),]
12 entrezID2symbol[match(my_symbol_gene,entrezID2symbol$symbol),]
```

## 10: [根据指定染色体及坐标得到序列](#)

---

### 题目

参考基因组hg19，指定染色体及坐标"chr5","8397384"，编写程序得到这个坐标以及它上下一个碱基。（机器无法计算hg19，则使用测试数据，指定坐标是 3号染色体的第6个碱基。）

### 测试数据

```
1 | >chr_1
2 | ATCGTCGaaAATGAANccNNttGTA
3 | AGGTCTNAAccAAttGggG
4 | >chr_2
5 | ATCGAATGATCGANNNGccTA
6 | AGGTCTNAAAAGG
7 | >chr_3
8 | ATCGTCGANNNGTAATggGA
9 | AGGTCTNAAAAGG
10 | >chr_4
11 | ATCGTCaaaGANNAATGANGgggTA
```

# 11： [根据指定染色体及坐标得到位置信息](#)

## 题目

基因的chr,start,end都是已知的（坐标是hg38系统），任意给定基因组的chr:pos(chr1:2075000-2930999), 判断该区间在哪个基因上面？（可用现成软件bedtools）

## 测试数据

|    |      |           |           |
|----|------|-----------|-----------|
| 1  | chr7 | 148697841 | 148698941 |
| 2  | chr7 | 148698942 | 148699029 |
| 3  | chr7 | 148699911 | 148701053 |
| 4  | chr7 | 148701109 | 148701307 |
| 5  | chr7 | 148701354 | 148702694 |
| 6  | chr7 | 148703100 | 148703520 |
| 7  | chr7 | 148703831 | 148704175 |
| 8  | chr7 | 148704484 | 148704734 |
| 9  | chr7 | 148704857 | 148705937 |
| 10 | chr7 | 148706271 | 148706671 |

## 12: [把文件内容按照染色体分开写出](#)

### 题目

根据染色体把文件拆分成1~22和其它染色体的两个文件呢？

### 测试数据

```
1 wget ftp://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_human/CCDS.current.txt
```

把文件改成bed格式，如下：

```
1 chr2 43995310 43995986
2 chr17 49788603 49789067
3 chr17 59565573 59566163
4 chr19 8390308 8390745
5 chr12 49188033 49189033
6 chr7 974903 975570
7 chr7 98878532 98879500
8 chr7 44044672 44045322
9 chr1 153634052 153634772
10 chr11 60905850 60906575
```

### Perl代码示例

```
1 use FileHandle;
2 foreach( 1..22 ){
3     $normal_chr{"chr".$_}=1 ;
4     open $fh{"chr".$_}, ">chr$_ .txt" or die;
5 }
6 open other, ">chr_other.txt" or die;
7 open FH, 'a.bed';
8 while(<FH>){
9     chomp;
10    @F=split;
11    if(exists $normal_chr{$F[0]}){
12        $fh{$F[0]}->print( "$_\n" );
13    }else{
14        print other $_;
15    }
16 }
17 foreach( 1..22 ){
18     close $fh{$_};
19 }
20 close other;
```



## 13: JSON格式数据的格式化

### 题目

学习json格式，下载测试数据，从该json文件里面提取：technique factor target principal\_investigator submission label category type Developmental-Stage organism key这几列信息。

### 测试数据

[http://biotraineer.com/jbrowse/JBrowse-1.12.1/sample\\_data/json/modencode/modencodeMetaData.json](http://biotraineer.com/jbrowse/JBrowse-1.12.1/sample_data/json/modencode/modencodeMetaData.json)

### Perl代码示例

```
1  #!/usr/bin/env perl
2  use strict;
3  use warnings;
4  use autodie ':all';
5  use 5.10.0;
6
7  use JSON 2;
8
9  my $data = from_json( do { local $/; open my $f, '<', $ARGV[0]; scalar <$f> } );
10
11  my @fields = qw( technique factor target principal_investigator submission label category
12                  type Developmental-Stage organism key );
13
14  say join ',', map "\"$_\"", @fields;
15
16  for my $item ( @{$data->{items}} ) {
17      $item->{key} = $item->{label};
18      no warnings 'uninitialized';
19      for my $track ( @{$item->{Tracks}} ) {
20          $item->{label} = $track;
21          say join ',', map "\"$_\"", @{$item}{@fields};
22      }
23  }
```

### 参考结果

完成之后的结果应该是：[http://biotraineer.com/jbrowse/JBrowse-1.12.1/sample\\_data/json/modencode/modencodeMetaData.csv](http://biotraineer.com/jbrowse/JBrowse-1.12.1/sample_data/json/modencode/modencodeMetaData.csv)

![11]

(([https://raw.githubusercontent.com/shenmengyuan/Python\\_Biotraineer/master/image/0.7400584788693336.png](https://raw.githubusercontent.com/shenmengyuan/Python_Biotraineer/master/image/0.7400584788693336.png))

## 14: 多个探针对应一个基因，取平均值、最大值

### 题目

编写脚本对多个探针对应一个基因，取平均值、最大值。

### 测试数据

### R代码示例

```
1 # 平均值
2 BiocInstaller::biocLite('CLL')
3 BiocInstaller::biocLite('hgu95av2.db')
4
5 library('hgu95av2.db')
6 library(CLL)
7 data(sCLLex)
8 sCLLex=sCLLex[,1:8] ## 样本太多，我就取前面8个
9 group_list=sCLLex$Disease
10 exprSet=exprs(sCLLex)
11 exprSet=as.data.frame(exprSet)
12 exprSet$probe_id=rownames(exprSet)
13 head(exprSet)
14 probe2symbol=toTable(hgu95av2SYMBOL)
15 dat=merge(exprSet,probe2symbol,by='probe_id')
16 results=t(sapply(split(dat,dat$symbol),function(x) colMeans(x[,1:(ncol(x)-1)])))
```

## 15: 把counts矩阵转换成RPKM矩阵

---

### 题目

编写脚本将hisat2+htseq-counts之后得到reads的counts矩阵转换成RPKM矩阵。

### 测试数据

```
1 wget ftp://ftp.ncbi.nlm.nih.gov/pub/CCDS/current_mouse/CCDS.current.txt
2 grep -v '^#' CCDS.current.txt | perl -alne '{/\[(..*?)\]/;$len=0;foreach(split/,,$1)
{@tmp=split/-/;$len+=($tmp[1]-$tmp[0])};$h{$F[2]}=$len if $len >$h{$F[2]}} END{print
"$_\t$h{$_}" foreach sort keys %h}' >mm10_ccds_length.txt
```

### R代码示例

```

1 genes_len=read.table("mm10_ccds_length.txt",stringsAsFactors=F)
2 head(genes_len)
3           V1      V2
4 1      -343C11.2  1139
5 2 00R_AC107638.2  1060
6 3      00R_Pgap2  1676
7 4 0610005C13Rik  7363
8 5 0610006L08Rik 34995
9 6 0610007P14Rik  9074
10
11
12 colnames(genes_len)<- c("GeneName", "Len")
13
14 head(exprSet)
15           GSM860181_priSG-A GSM860182_SG-A GSM860183_SG-B GSM860184_lepSC
16 00R_AC107638.2           0           1           0           1
17 0610005C13Rik           20           22           11           27
18 0610006L08Rik           0           0           0           2
19 0610007P14Rik          2075          1785          1269          1430
20 0610009B22Rik           256           376           300           386
21 0610009E02Rik           22           22           16           28
22
23 exprSet<-exprSet[ rownames(exprSet) %in% genes_len$GeneName ,]
24
25 total_count<- colSums(exprSet)
26
27 neededGeneLength=genes_len[ match(rownames(exprSet), genes_len$GeneName) ,2]
28
29 rpkm <- t(do.call( rbind,lapply(1:length(total_count),function(i){
30     10^9*exprSet[,i]/neededGeneLength/total_count[i]
31 })))
32 head(rpkm)
33 rownames(rpkm)=rownames(exprSet)
34 colnames(rpkm)=colnames(exprSet)
35
36 write.table(rpkm,file="rpkm.txt",sep="\t",quote=F)
37
38
39
40 library(TxDb.Mmusculus.UCSC.mm10.knownGene)
41 txdb=TxDb.Mmusculus.UCSC.mm10.knownGene
42 gt=transcriptsBy(txdb,by="gene")
43 lapply(gt[1:40],function(x) max(width(x)))
44 library(org.Mm.eg.db)
45
46 mykeys=
47 columns(txdb);keytypes(txdb)
48 neededcols <- c("GENEID", "TXSTRAND", "TXCHROM")
49 select(txdb, keys=mykeys, columns=neededcols, keytype="TXNAME")

```

## 参考结果

## 16: 对有临床信息的表达矩阵批量做生存分析

---

### 题目

使用R实现生存分析：用 `my.surv <- surv(OS_MONTHS, OS_STATUS=='DECEASED')` 构建生存曲线；用 `kmfit2 <- survfit(my.surv~TUMOR_STAGE_2009)` 来做某一个因子的KM生存曲线；用 `survdiff(my.surv~type, data=dat)` 来看看这个因子的不同水平是否有显著差异，其中默认用的是logrank test 方法；用 `coxph(Surv(time, status) ~ ph.ecog + tt(age), data=lung)` 来检测自己感兴趣的因子是否受其它因子(age,gender等等)的影响。

### 代码示例

```

1 rm(list=ls())
2 ## 50 patients and 200 genes
3 dat=matrix(rnorm(10000,mean=8,sd=4),nrow = 200)
4 rownames(dat)=paste0('gene_',1:nrow(dat))
5 colnames(dat)=paste0('sample_',1:ncol(dat))
6 os_years=abs(floor(rnorm(ncol(dat),mean = 50,sd=20)))
7 os_status=sample(rep(c('LIVING', 'DECEASED'),25))
8
9 library(survival)
10 my.surv <- Surv( os_years,os_status=='DECEASED')
11 ## The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE =
12 death) or 1/2 (2=death).
13 ## And most of the time we just care about the time od DECEASED .
14
15 fit.KM=survfit(my.surv~1)
16 fit.KM
17 plot(fit.KM)
18
19 log_rank_p <- apply(dat, 1, function(values1){
20   group=ifelse(values1>median(values1),'high','low')
21   kmfit2 <- survfit(my.surv~group)
22   #plot(kmfit2)
23   data.survdiff=survdiff(my.surv~group)
24   p.val = 1 - pchisq(data.survdiff$chisq, length(data.survdiff$n) - 1)
25 })
26 names(log_rank_p[log_rank_p<0.05])
27
28 gender= ifelse(rnorm(ncol(dat))>1,'male','female')
29 age=abs(floor(rnorm(ncol(dat),mean = 50,sd=20)))
30 ## gender and age are similar with group(by gene expression)
31
32 cox_results <- apply(dat , 1, function(values1){
33   group=ifelse(values1>median(values1),'high','low')
34   survival_dat <- data.frame(group=group,gender=gender,age=age,stringsAsFactors = F)
35   m=coxph(my.surv ~ age + gender + group, data = survival_dat)
36
37   beta <- coef(m)
38   se <- sqrt(diag(vcov(m)))
39   HR <- exp(beta)
40   HRse <- HR * se
41
42   #summary(m)
43   tmp <- round(cbind(coef = beta, se = se, z = beta/se, p = 1 - pchisq((beta/se)^2, 1),
44     HR = HR, HRse = HRse,
45     HRz = (HR - 1) / HRse, HRp = 1 - pchisq(((HR - 1)/HRse)^2, 1),
46     HRCILL = exp(beta - qnorm(.975, 0, 1) * se),
47     HRCIUL = exp(beta + qnorm(.975, 0, 1) * se)), 3)
48   return(tmp['grouplow',])
49 })
50 cox_results[,cox_results[4,]<0.05]

```

PS: 里面的os\_years应该修改为os\_month，因为总的生存期不应该长达几十年，而且因为ag和os\_years都是随机生成的，可能会出现很不符合自然科学的现象。但是这个模拟数据，请大家不用较真。

## 17: [对多个差异分析结果直接取交集并集](#)

### 题目

编写脚本对每两个差异分析结果计算基因交集个数与基因并集个数的比值，得到一个比值矩阵。

### 测试数据

用perl单行命令模拟数据：

```
1 perl -e 'BEGIN{use List::Util qw/shuffle/;@gene=A..Z}{foreach(1..10)
  {@this_genes=@gene[(shuffle(0..25))[0..int(rand(20))+4]];print "comparison_$_ <- ";print
  join(";",@this_genes);print "\n"}}'
```

总共10次差异分析，每次差异分析得到的差异基因在同一行的后面用大小字母表示。

```
1 comparison_1 -> I;G;E;V;C;K;B;W
2 comparison_2 -> G;E;N;H;Y;M;L;S;K;A;J;O;D;P;R;U;Q;F;Z;C
3 comparison_3 -> Y;V;U;N;H;K;I;P;S;F;D;X;G;C;Z;J;Q;T;W;O;E;M
4 comparison_4 -> N;T;K;B;H;Z;W;C;Q;I;V;F;D;S;R;Y;J;X;P;O;G;L;A
5 comparison_5 -> G;J;A;H;W;T;Z;E;Y;S;R
6 comparison_6 -> Z;M;D;R;P;G;L;W;Y;U;X;E;A;S;T;I;H
7 comparison_7 -> H;Z;T;O;W;Q;M;X;J;N;U;K;F;P;I;C;S;Y;A;B
8 comparison_8 -> A;R;L;T;W;Q;S;F;P;X;E;V;Y;G;K;J;Z;C
9 comparison_9 -> J;X;K;D;O;H;L;F;C;P;R;N
10 comparison_10 -> G;S;K;H;C;O;W;F;Q;X
```

### R代码示例

```
1 a=readLines('test.txt')
2 n=unlist(lapply(a , function(x){
3   trimws(strsplit(x,'<-')[[1]][1])
4 })))
5 v=lapply(a , function(x){
6   trimws(strsplit(trimws(strsplit(x,'<-')[[1]][2]),';')[[1]])
7 })
8 names(v)=n
9 tmp=unlist(lapply(v, function(x){
10   lapply(v, function(y){
11     p1=length(intersect(x,y))
12     p2=length(union(x,y))
13     return(p1/p2)
14   })
15 })))
16 out=matrix(tmp,nrow = length(n))
17 rownames(out)=n
18 colnames(out)=n
19 out[1:5,1:5]
```



## 参考结果

结果的前五行如下：

|   |              |              |              |              |              |           |
|---|--------------|--------------|--------------|--------------|--------------|-----------|
| 1 | comparison_1 | comparison_2 | comparison_3 | comparison_4 | comparison_5 |           |
| 2 | comparison_1 | 1.0000000    | 0.1666667    | 0.3043478    | 0.2916667    | 0.1875000 |
| 3 | comparison_2 | 0.1666667    | 1.0000000    | 0.6800000    | 0.6538462    | 0.4090909 |
| 4 | comparison_3 | 0.3043478    | 0.6800000    | 1.0000000    | 0.7307692    | 0.3750000 |
| 5 | comparison_4 | 0.2916667    | 0.6538462    | 0.7307692    | 1.0000000    | 0.4166667 |
| 6 | comparison_5 | 0.1875000    | 0.4090909    | 0.3750000    | 0.4166667    | 1.0000000 |

## 18: [根据GTF格式的基因注释文件得到人所有基因的染色体坐标](#)

### 题目

从gencode数据库里面可以下载所有的gtf文件，编写脚本得到基因的染色体、起始终止坐标如下：

```
1 [jianmingzeng@gencode]$ head protein_coding.hg19.position
2 chr1      69091      70008      OR4F5
3 chr1      367640     368634     OR4F29
4 chr1      621096     622034     OR4F16
5 chr1      859308     879961     SAMD11
6 chr1      879584     894689     NOC2L
7 chr1      895967     901095     KLHL17
8 chr1      901877     911245     PLEKHN1
9 chr1      910584     917473     PERM1
10 chr1     934342     935552     HES4
11 chr1     936518     949921     ISG15
12 [jianmingzeng@gencode]$ head protein_coding.hg38.position
13 chr1      69091      70008      OR4F5
14 chr1     182393     184158     F0538757.2
15 chr1     184923     200322     F0538757.1
16 chr1     450740     451678     OR4F29
17 chr1     685716     686654     OR4F16
18 chr1     923928     944581     SAMD11
19 chr1     944204     959309     NOC2L
20 chr1     960587     965715     KLHL17
21 chr1     966497     975865     PLEKHN1
22 chr1     975204     982093     PERM1
```