基因课 R 语言基础专题教材

张旭东、王莹

2020-06-08

很多事情从你决定开始的一瞬间起

最困难的时刻就已经过去了

目录

前言

关于基因课

成立于 2017 年 7 月，专注于生物信息学培训。

基因课网校：已完成 14 门视频课程的开发，累计学员 5000 余人。

线下培训班：武汉、成都、西安、杭州、广州、北京等地举办快速入门班 30 余期。

基因课网站：www.genek.cn

关于作者

主讲老师张旭东

曾任华大基因项目组组长、产品研发组组长。2014 年合伙创办贝纳基因，担任 CTO。2017 年创办基因课，专注生物信息学培训。

发表文章：麻疯树基因组计划（the plant journal）、鹰嘴豆基因组计划（Nature Biotechnology）、芝麻基因组计划（Genome Biology）、猕猴桃群体研究（New Phytologist）。

主讲老师王莹

曾任华大基因、贝纳基因高级生物信息工程师。

擅长基因组拼接、比较基因组和进化分析、重测序分析。

发表文章：甲藻基因组（Science）、穿心莲基因组（The Plant Journal）、野生大豆基因组（Nature Communications）、菊花基因组（Molecular

Plant）、鹰嘴豆基因组计划（Nature Biotechnology）、芝麻基因组计划（Genome Biology）、猕猴桃群体研究（New Phytologist）。

第一章　R 基础

1.1　环境部署

R 语言可以在各个操作系统上运行，如果个人电脑配置较高，可以在自己电脑上安装，也可以使用我们服务器上安装好的。服务器 Rstudio 登录地址：gs0.genek.tv:8787

1.1.1　安装 R

请到 https://cran.rstudio.com/index.html 下载 R 安装包，一路下一步，即可安装。

1.1.2　安装 Rstudio

登录 https://www.rstudio.com/products/rstudio/download/，下载 Rstudio 桌面版。安装过程中一直下一步即可。

1.1.3　安装软件包

1.1.3.1　注意事项

- 来自 cran 的软件包

```
options(repos="http://mirrors.tuna.tsinghua.edu.cn/CRAN/")
install.packages("dplyr")
```

- 来自 bioconductor 的软件包

```
options(BioC_mirror="http://mirrors.tuna.tsinghua.edu.cn/bioconductor/")
BiocManager::install("DESeq2")
```

1.1.4 设置工作目录

获取当前工作目录

```
getwd()
```

```
## [1] "/home/zhxd/workspace/R_basic"
```

设置工作目录

```
setwd("/home/zhxd/My_Course/BioR")
```

如果你使用的是 Rstudio，建议创建一个 Rstudio Project，这样工作目录默认就是项目目录。

1.2 快速上手

我们先演示一个表达数据分析的例子，包括导入数据、数据过滤、数据变形和可视化。

假设各个基因在各个样本中的表达量如下

```
gene1_s1 <- 10
gene1_s2 <- 20
gene1_s3 <- 30
gene1_s4 <- 40
```

求 gene1 在各个样本中表达量之和

```
gene1_total <- gene1_s1 + gene1_s2 + gene1_s3 + gene1_s4
gene1_total
```

```
## [1] 100
```

我们把单个元素的变量叫做**标量**，如果有几十个样本，使用标量效率太低，就需要定义**向量**

```
gene1 <- c(10, 20, 30, 40)
gene1
```

```
## [1] 10 20 30 40
```

求 gene1 在各个样本中表达量之和有多种办法

方法一：**循环**

每次取出一条基因，累加

```
gene1_total <- 0
for (i in gene1){
  gene1_total = gene1_total + i
}
gene1_total
```

```
## [1] 100
```

方法二：调用**函数**

直接调用别人写好的求和函数

```
gene1_total <- sum(gene1)
gene1_total
```

```
## [1] 100
```

如果有几万条基因、几十个样本呢？就需要定义一个二维的数据结构，类似 Excel 表。这在 R 中可以用 data.frame。

读取文件

```
gene_exp <- read.delim("data/R_basic/genes.TMM.EXPR.matrix", row.names=1)
```

要求每一条基因表达量的和，可以对每一行进行循环，但太麻烦了，R 语言有自己的想法

```
head(apply(gene_exp, 1, sum))
```

```
##  HF36249  HF06454  HF31517  HF07207  HF16138  HF24764
##    0.000    5.231 1074.661    0.000    0.000  786.410
```

也可以求每个样本中所有基因表达量的和

```
head(apply(gene_exp, 2, sum))
```

```
## BLO_S1_LD1 BLO_S1_LD2 BLO_S1_LD3 BLO_S2_LD1 BLO_S2_LD2
##     886715     923047     898468    1041709     999279
## BLO_S2_LD3
##     950409
```

增加一个需求，求样本之间的相关系数

```
sample_cor <- cor(gene_exp)
sample_cor[1:4,1:4]
```

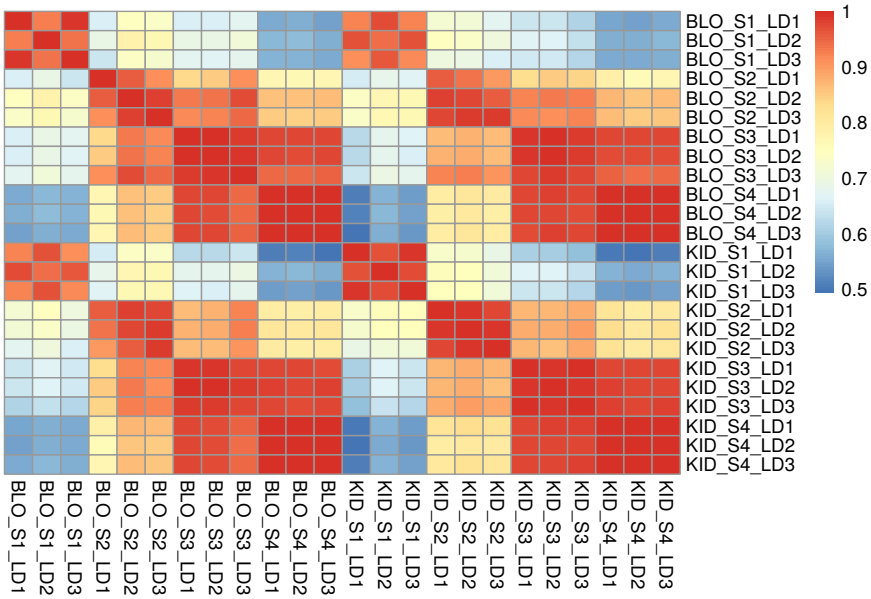
```
##             BLO_S1_LD1 BLO_S1_LD2 BLO_S1_LD3 BLO_S2_LD1
## BLO_S1_LD1     1.0000     0.9244     0.9907     0.6604
## BLO_S1_LD2     0.9244     1.0000     0.9391     0.6904
## BLO_S1_LD3     0.9907     0.9391     1.0000     0.6455
## BLO_S2_LD1     0.6604     0.6904     0.6455     1.0000
```


画个 heatmap，这时候就需要调用别人写好的**软件包**

```
# 安装 pheatmap 包
install.packages("pheatmap")
```

```
# 加载 pheatmap 包
library(pheatmap, quietly = T)

# 调用 pheatmap 包里的 pheatmap() 函数画图
pheatmap(sample_cor,
         cluster_rows = F,
         cluster_cols = F)
```



所以，学习一门编程语言要学习

- 怎么读取数据？输入输出
- 怎么保存变量？数据结构，如标量、向量、数据框
- 保存成数字还是字符？变量类型
- 怎么进行运算？计算、函数、流程控制
- 怎么进行绘图？

1.3 数据结构

1.3.1 原子类型

向量是 R 语言的原子类型。

Python 的原子数据类型是标量

```
a = 1
b = [1]
a == b
```

```
## False
```

而 R 的原则数据类型是向量

```
a <- 1
b <- c(1)
a == b
```

```
## [1] TRUE
```

1.3.2 向量 Vector

1.3.2.1 创建

创建一个包含 5 个元素的向量 v

```
v <- c(1, 2, 3, 4, 5)
v
```

```
## [1] 1 2 3 4 5
```

查看数据类型

```
class(v)
```

```
## [1] "numeric"
```

快速创建连续的整数向量

```
1:5
```

```
## [1] 1 2 3 4 5
```

快速创建重复的向量

```
rep(3,10)
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3
```

1.3.2.2 取值

向量的索引也是一个向量

1.3.2.2.1 按位置

向量的位置是从 1 开始的


```
v[1]
```

```
## [1] 1
```

```
v[1:3]
```

```
## [1] 1 2 3
```

```
v[c(1,3,5)]
```

```
## [1] 1 3 5
```

```
v[-c(2,4)]
```

```
## [1] 1 3 5
```

1.3.2.2.2 按逻辑

```
v[v%%2 == 0]
```

```
## [1] 2 4
```

1.3.2.2.3 按名称

可以为向量命名

```
names(v) <- c('chr1', 'chr2', 'chr3', 'chr4', 'chr5')
v
```

```
## chr1 chr2 chr3 chr4 chr5
##    1    2    3    4    5
```

```
v['chr3']
```

```
## chr3
##    3
```

1.3.2.3 计算

```
v + 1
```

```
## chr1 chr2 chr3 chr4 chr5
##    2    3    4    5    6
```

```
v + c(1,2)
```

```
## chr1 chr2 chr3 chr4 chr5
##    2    4    4    6    6
```

两个向量相加时，会自动扩展成一样长度

1.3.3 数据框 DataFrame

1.3.3.1 创建

使用 data.frame 创建数据框，数据框的每一列叫做一个变量，每一行叫做一条观测。


```
df <- data.frame(S1 = c(1, 2, 3, 4, 5),
                 S2 = c(5, 4, 3, 2, 1),
                 S3 = runif(5, -10, 10), # -10 到 10 之间，生成 5 个数
                 S4 = runif(5, -10, 10))
df
```

```
##   S1 S2      S3     S4
## 1  1  5  0.3216 -2.796
## 2  2  4  7.7613  2.929
## 3  3  3 -4.0966 -7.594
## 4  4  2  0.7800 -2.796
## 5  5  1 -8.7360 -5.057
```

查看列名

```
colnames(df)
```

```
## [1] "S1" "S2" "S3" "S4"
```

查看行名，默认的列名是行数

```
rownames(df)
```

```
## [1] "1" "2" "3" "4" "5"
```

可以使用 rownames 这是行名

```
rownames(df) <- c("gene1", "gene2", "gene3", "gene4", "gene5")
df
```

```
##       S1 S2      S3     S4
## gene1  1  5  0.3216 -2.796
## gene2  2  4  7.7613  2.929
## gene3  3  3 -4.0966 -7.594
## gene4  4  2  0.7800 -2.796
## gene5  5  1 -8.7360 -5.057
```


列数

```
ncol(df)
```

```
## [1] 4
```

行数

```
nrow(df)
```

```
## [1] 5
```

行数和列数

```
dim(df)
```

```
## [1] 5 4
```

查看前 2 行

```
head(df, n = 2)
```

```
##       S1 S2     S3     S4
## gene1  1  5 0.3216 -2.796
## gene2  2  4 7.7613  2.929
```

后 2 行

```
tail(df, n = 2)
```

```
##       S1 S2     S3     S4
## gene4  4  2  0.780 -2.796
## gene5  5  1 -8.736 -5.057
```

1.3.3.2 取值

1.3.3.2.1 提取变量

返回一个向量

```
df$S1
```

```
## [1] 1 2 3 4 5
```

1.3.3.2.2 按位置

```
df[1:3,1:3]
```

```
##        S1 S2      S3
## gene1   1  5  0.3216
## gene2   2  4  7.7613
## gene3   3  3 -4.0966
```

1.3.3.2.3 按名称

```
df[c('gene1', 'gene3'),c('S1', 'S2' )]
```

```
##        S1 S2
## gene1   1  5
## gene3   3  3
```

留空则代表所有行或列

```
df[,1:3]
```

```
##        S1 S2      S3
## gene1   1  5  0.3216
## gene2   2  4  7.7613
## gene3   3  3 -4.0966
## gene4   4  2  0.7800
## gene5   5  1 -8.7360
```

1.3.3.2.4 按逻辑

筛选 S1 中表达大于 1 的基因，只显示 S2 和 S4 两列

```
subset(df, S1>1, select = c("S2", "S4"))
```

```
##       S2     S4
## gene2  4  2.929
## gene3  3 -7.594
## gene4  2 -2.796
## gene5  1 -5.057
```

1.3.3.2.5 提取元素

返回向量

```
df[2,2]
```

```
## [1] 4
```


1.3.3.3 计算

1.3.3.3.1 与向量计算

```
head(df + c(1,2))
```

```
##       S1 S2     S3      S4
## gene1  2  7  1.322 -0.7963
## gene2  4  5  9.761  3.9288
## gene3  4  5 -3.097 -5.5939
## gene4  6  3  2.780 -1.7960
## gene5  6  3 -7.736 -3.0574
```

1.3.3.3.2 按行、列计算

```
rowSums(df)
```

```
##  gene1  gene2  gene3  gene4  gene5
##  3.525 16.690 -5.691  3.984 -7.793
```

```
colSums(df)
```

```
##     S1     S2     S3     S4
##  15.00  15.00  -3.97 -15.31
```

```
colMeans(df)
```

```
##     S1     S2     S3     S4
##  3.0000  3.0000 -0.7939 -3.0629
```



```
rowMeans(df)
```

```
##   gene1   gene2   gene3   gene4   gene5
##  0.8813  4.1725 -1.4226  0.9960 -1.9483
```

更多功能参考 matrixStats 包：https://cran.r-project.org/web/packages/matrixStats/vignettes/matrixStats-methods.html

1.3.3.3.3 apply 函数

也可以使用 apply 函数按行或者列进行计算

```
apply(df, 1, mean)
```

```
##    gene1    gene2    gene3    gene4    gene5
##   0.8813   4.1725 -1.4226   0.9960 -1.9483
```

```
apply(df, 2, mean)
```

```
##       S1       S2       S3       S4
##   3.0000   3.0000 -0.7939 -3.0629
```

1.4 数据类型

1.4.1 数值型 numeric

1.4.1.1 定义


```
a <- c(1, -3, 5.64)
b <- c(1, 4, 2.3)
```

```
class(a)
```

```
## [1] "numeric"
```

1.4.1.2 计算

四舍五入到指定位小数

```
round(a, digits = 1)
```

```
## [1]  1.0 -3.0  5.6
```

绝对值

```
abs(a)
```

```
## [1] 1.00 3.00 5.64
```

开平方

```
sqrt(b)
```

```
## [1] 1.000 2.000 1.517
```

求和

```
sum(a)
```

```
## [1] 3.64
```


求标准差

```
sd(a)
```

```
## [1] 4.324
```

相乘

```
a * b
```

```
## [1]   1.00 -12.00  12.97
```

求幂

```
a ** b
```

```
## [1]  1.00 81.00 53.45
```

```
# a^b
```

相除

```
a / b
```

```
## [1]  1.000 -0.750  2.452
```

取余

```
a %% b
```

```
## [1] 0.00 1.00 1.04
```

整除

```
a %/% b
```

```
## [1]  1 -1  2
```

求相关系数

```
cor(a, b)
```

```
## [1] -0.5292
```

1.4.2 字符型 character

1.4.2.1 定义

```
s <- c('A', 'TT', 'CCC', 'GGGG')
s
```

```
## [1] "A"    "TT"   "CCC"  "GGGG"
```

查看变量的数据类型

```
class(s)
```

```
## [1] "character"
```


1.4.2.2 计算

R 语言自带的字符串处理函数不如 stringr 包中的好用。

```
library(stringr, quietly = T)
```

计数

```
str_count(s, 'T')
```

```
## [1] 0 2 0 0
```

按位置切分

```
str_sub(s, 1, 3)
```

```
## [1] "A"   "TT"  "CCC" "GGG"
```

包含 'C' 的元素

```
str_subset(s, 'C')
```

```
## [1] "CCC"
```

求长度

```
str_length(s)
```

```
## [1] 1 2 3 4
```


替换

```
str_replace(s, 'G', 'N')
```

```
## [1] "A"    "TT"   "CCC"  "NGGG"
```

替换所有

```
str_replace_all(s, 'G', 'N')
```

```
## [1] "A"    "TT"   "CCC"  "NNNN"
```

转小写

```
str_to_lower(s)
```

```
## [1] "a"    "tt"   "ccc"  "gggg"
```

连接

```
str_c(c('chr1', 'chr2', 'chr3', 'chr4'), s, sep = ':')
```

```
## [1] "chr1:A"    "chr2:TT"   "chr3:CCC"  "chr4:GGGG"
```

拆分

```
str_split(s, '', simplify = T)
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "A"  ""   ""   ""
## [2,] "T"  "T"  ""   ""
## [3,] "C"  "C"  "C"  ""
## [4,] "G"  "G"  "G"  "G"
```

排序

```
str_sort(s)
```

```
## [1] "A"    "CCC"  "GGGG" "TT"
```

1.4.3 逻辑型 logical

```
x <- c(T, F, T)
y <- c(F, F, F)
class(x)
```

```
## [1] "logical"
```

或

```
x | y
```

```
## [1]  TRUE FALSE  TRUE
```



```
x || y
```

```
## [1] TRUE
```

与

```
x & y
```

```
## [1] FALSE FALSE FALSE
```

```
x && y
```

```
## [1] FALSE
```

非

```
!x
```

```
## [1] FALSE  TRUE FALSE
```

所有都为真?

```
all(x)
```

```
## [1] FALSE
```

至少有一个为真?

```
any(x)
```

```
## [1] TRUE
```


1.5 输入输出

1.5.1 输入输出数据

1.5.1.1 输入

```
gene_exp <- read.table("data/R_basic/genes.TMM.EXPR.matrix",
                       header = T,
                       row.names=1)
gene_exp[1:3,1:3]
```

```
##          BLO_S1_LD1 BLO_S1_LD2 BLO_S1_LD3
## HF36249         0.0       0.00       0.00
## HF06454         0.0       0.00       0.00
## HF31517        54.1      54.63      72.65
```

1.5.1.2 输出

```
write.table(gene_exp,
            file = "output/gene_exp.txt",
            sep = "\t",
            quote = F)
```

1.5.2 保存加载对象

1.5.2.1 保存

```
a <- 1
b <- 2
save(a, b, file = 'output/mydata.rdata')
rm(a, b)
```


1.5.2.2 加载

```
load('output/mydata.rdata')
```

1.5.3 保存图片

```
pdf(file = 'output/myplot.pdf')
plot(x = runif(n = 10, 1, 100), y = runif(n = 10, 1, 100))
dev.off()
```

```
## cairo_pdf
##         2
```

第二章　R 进阶

2.1　准备

```
library(tidyverse)
```

2.2　数据类型 Tibble

DataFrame 是 R 语言的基础数据结构，Tibble 是 Hadley 开发的用于取代 data.frame 的新数据类型。

- data.frame 与 tibble 的区别

```
# data.frame
gene_exp_df <- read.delim("data/R_basic/gene_exp.txt", row.names=1)

# tibble
library(readr)
gene_exp_tbl <- read_delim("data/R_basic/gene_exp.txt",
    "\t", escape_double = FALSE, trim_ws = TRUE)
```

- 将 data.frame 转换为 tibble

```
library(tibble)
as_tibble(rownames_to_column(gene_exp_df, var = "gene_id"))
```

```
## # A tibble: 5 x 5
##   gene_id sample1 sample2 sample3 sample4
##   <chr>     <int>   <int>   <int>   <int>
## 1 gene1         3       5     287      41
## 2 gene2       120      NA      33      31
## 3 gene3        39      12      34      33
## 4 gene4        23      22       0      78
## 5 gene5        23      32      49       3
```

- 将 tibble 转换为 data.frame

```
as.data.frame(column_to_rownames(gene_exp_tbl, var = "gene_id"))
```

```
##       sample1 sample2 sample3 sample4
## gene1       3       5     287      41
## gene2     120      NA      33      31
## gene3      39      12      34      33
## gene4      23      22       0      78
## gene5      23      32      49       3
```

2.3 数据处理 Dplyr

2.3.1 导入测试数据

```
library(readr)
de_result <- read_delim("data/R_basic/de_result.txt",
    "\t", escape_double = FALSE, trim_ws = TRUE)
```

2.3.2 select 选择行

select 函数用于选择列

```
dplyr::select(de_result, gene_id, logFC, padj = FDR)
```

```
## # A tibble: 5 x 3
##   gene_id logFC    padj
##   <chr>   <dbl>   <dbl>
## 1 gene1     3   0.03
## 2 gene2     1   0.38
## 3 gene3     2   0.0025
## 4 gene4    -3   0.003
## 5 gene5    -0.4 0.00480
```

也可以排除法

```
dplyr::select(de_result, -pvalue)
```


2.3.3 mutate 添加列

mutate 函数用于添加一列

```
mutate(de_result, FC = 2 ** logFC)
```

```
## # A tibble: 5 x 5
##   gene_id logFC pvalue     FDR    FC
##   <chr>   <dbl>  <dbl>   <dbl> <dbl>
## 1 gene1     3    0.01  0.03    8
## 2 gene2     1    0.3   0.38    2
## 3 gene3     2    0.002 0.0025  4
## 4 gene4    -3    0.002 0.003   0.125
## 5 gene5    -0.4  0.004 0.00480 0.758
```

2.3.4 filter 选择行

filter 函数用于选择行，& 表示与，| 表示或

```
filter(de_result, abs(logFC) >= 1 & FDR < 0.05)
```

```
## # A tibble: 3 x 4
##   gene_id logFC pvalue    FDR
##   <chr>   <dbl>  <dbl>  <dbl>
## 1 gene1       3  0.01   0.03
## 2 gene3       2  0.002 0.0025
## 3 gene4      -3  0.002 0.003
```

删除缺失数据

```
filter(gene_exp_tbl, !is.na(sample2))
```

```
## # A tibble: 4 x 5
##   gene_id sample1 sample2 sample3 sample4
##   <chr>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 gene1         3       5     287      41
## 2 gene3        39      12      34      33
## 3 gene4        23      22       0      78
## 4 gene5        23      32      49       3
```

2.3.5 arrange 排序

按 FDR 升序排列

```
arrange(de_result, FDR)
```

```
## # A tibble: 5 x 4
##   gene_id logFC pvalue     FDR
##   <chr>   <dbl>  <dbl>   <dbl>
## 1 gene3     2    0.002 0.0025
## 2 gene4    -3    0.002 0.003
## 3 gene5    -0.4  0.004 0.00480
## 4 gene1     3    0.01  0.03
## 5 gene2     1    0.3   0.38
```

按 FDR 降序排列

```
arrange(de_result, desc(FDR))
```

```
## # A tibble: 5 x 4
##   gene_id logFC pvalue     FDR
##   <chr>   <dbl>  <dbl>   <dbl>
## 1 gene2     1    0.3   0.38
## 2 gene1     3    0.01  0.03
## 3 gene5    -0.4  0.004 0.00480
## 4 gene4    -3    0.002 0.003
## 5 gene3     2    0.002 0.0025
```

2.3.6 关联

```
library(readr)
gene_function <- read_delim("data/R_basic/gene_function.txt",
    "\t", escape_double = FALSE, trim_ws = TRUE)
```

```
left_join(de_result, gene_function, by = c('gene_id' = 'gene_name'))
```

```
## # A tibble: 5 x 5
##   gene_id logFC pvalue     FDR annotation
##   <chr>   <dbl>  <dbl>   <dbl> <chr>
## 1 gene1       3   0.01  0.03    aaa
## 2 gene2       1   0.3   0.38    bbb
## 3 gene3       2   0.002 0.0025  ccc
## 4 gene4      -3   0.002 0.003   <NA>
## 5 gene5      -0.4 0.004 0.00480 ddd
```

left_join：以左边的表为基准

right_join: 以右边的表为基准

full_join: 并集

inner_join: 交集

2.3.7 管道

```
dplyr::select(de_result, gene_id,  logFC, padj = FDR) %>%
  filter(abs(logFC) > 1 &  padj < 0.05) %>%
  left_join(gene_function, by = c('gene_id' = 'gene_name'))
```

```
## # A tibble: 3 x 4
##   gene_id logFC   padj annotation
##   <chr>   <dbl>  <dbl> <chr>
## 1 gene1       3 0.03   aaa
## 2 gene3       2 0.0025 ccc
## 3 gene4      -3 0.003  <NA>
```

练习：为表达矩阵添加差异 foldchange、annotation