L02 统计学导论 B-实验课 02

郑盼盼

2024-09-18

目录

2.0 变量赋值

- **变量名**：变量名仅包含任何大小写字母，数字以及下划线 _（但数字和下划线不能用于变量名开头）；且要注意区分大小写 a ≠ A，比如，如下的语句就会报错

```
a <- 1
A
```

- 三个赋值符号：

  1. `<-` (在 RStudio 中可以使用快捷键 `alt/option + -` 快速插入)
  2. `=`
  3. `->`

- 以将变量 `a` 赋值为 `1` 为例，以下三种赋值方式等价

```r
a <- 1
a = 1   # 注意：变量名必须在等号的左侧，而值在等号的右侧
1 -> a
```

2.1 变量类型

R 语言中变量类型主要包括逻辑型，数值型和字符型三类

可以使用 `class(变量名)` 的方式查看变量的类型

2.1.1 逻辑型 (logical)

- 逻辑型 (logical)：即对于“对 (True)”或“错 (False)”的表述。仅有 `TRUE`（可简写为 `T`）和 `FALSE`（可简写为 `F`）两类。

```r
# l1,l2,l3, l4 均为逻辑型变量（这里第一个是字母"l" 不是数字"1"
l1 <- TRUE
l2 <- FALSE
l3 <- T  # 等价于 l1
l4 <- F  # 等价于 l2

paste("l1 = ", l1, ", l1 is ", class(l1))
```

```
## [1] "l1 =  TRUE , l1 is  logical"
```

```r
paste("l2 = ", l2, ", l2 is ", class(l2))
```

```
## [1] "l2 =  FALSE , l2 is  logical"
```

```r
paste("l3 = ", l3, ", l3 is ", class(l3))
```

```
## [1] "l3 =  TRUE , l3 is  logical"
```

```r
paste("l4 = ", l4, ", l4 is ", class(l4))
```

```
## [1] "l4 =  FALSE , l4 is  logical"
```

- 试试看下面三个语句的结果，并判断结果是什么类型的变量

  1. `1 > 2`
  2. `1 == 0`
  3. `1 != 0`

- 我们可以使用 `is.logical(变量)` 的方式判断一个变量是否为逻辑型

```r
x <- 1 > 2
is.logical(x)
```

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

```
is.logical(1 > 2)
```

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

2.1.2 数值型 (numeric)

- 数值型 (numeric)：例如人的身高，体重，学生的成绩。我们可以用一个数值来表述这些性质。

```
# n1, n2, n3, n4 均为数值型变量。R 语言默认存储为浮点型
n1 <- 3
n2 <- 1/3
n3 <- 0.33333
n4 <- 3.3333e-1   # 科学计数法：3.3333e-1 等同于 3.3333 * (10^-1)

paste("n1 = ", n1,", n1 is the ", class(n1))
```

```
## [1] "n1 =  3 , n1 is the  numeric"
```

```
paste("n2 = ", n2,", n2 is the ", class(n2))
```

```
## [1] "n2 =  0.333333333333333 , n2 is the  numeric"
```

```
paste("n3 = ", n3,", n3 is the ", class(n3))
```

```
## [1] "n3 =  0.33333 , n3 is the  numeric"
```

```
paste("n4 = ", n4,", n4 is the ", class(n4))
```

```
## [1] "n4 =  0.33333 , n4 is the  numeric"
```

- 试试看：

1. `TRUE + TRUE`
2. `TRUE - FALSE`
3. `3 + FALSE`

2.1.3 字符型 (character)

- 字符型 (character)：例如人的性别 (gender)，姓名。一般我们使用一串字符来表示这些性质，字符型变量两边由 " 或 ' 包裹。

```
c1 = "TRUE"
c2 = "FALSE"
c3 = "this is 'test'"    # 若希望在字符串中加'可以在外侧用"
c4 = 'this is \'test\''  # 也可以在'前加\实现字符的转意
print(c1)
```

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

```
print(c2)
```

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

```
print(c3)
```

```
## [1] "this is 'test'"
```

```
print(c4)
```

```
## [1] "this is 'test'"
```

- 试试看

1. 'this is "test"'
2. "this is \"test\""

2.1.4 不同类型之间的相互转换

- 转化为数值型 as.numeric()

```
a <- "123"
as.numeric(a)
```

```
## [1] 123
```

```
b <- T
as.numeric(b)
```

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

- 转化为字符型 as.character()

```
a <- T
as.character(a)
```

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

```
b <- 1e-3
as.character(b)
```

```
## [1] "0.001"
```

- 转化为逻辑型 as.logical() 一切非零的数值都会被转化为 TRUE，而零会被转化为 FALSE

```
a <- 1
as.logical(a)
```

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

```r
b <- -1
as.logical(b)
```

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

```r
c <- 0
as.logical(c)
```

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

- 对于无法转换的类型，R 会自动转化为 NA 即缺失值

```r
a <- "0"
as.logical(a)
```

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

```r
b <- "1this is characters"
as.numeric(b)
```

```
## Warning: NAs introduced by coercion
```

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

2.1.5 特殊常量

变量名	含义
pi	圆周率，3.1415...
Inf	无穷大 1/0
NaN	不定量，0/0
NA	缺失值

```r
pi
```

```
## [1] 3.141593
```

```r
1/0
```

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

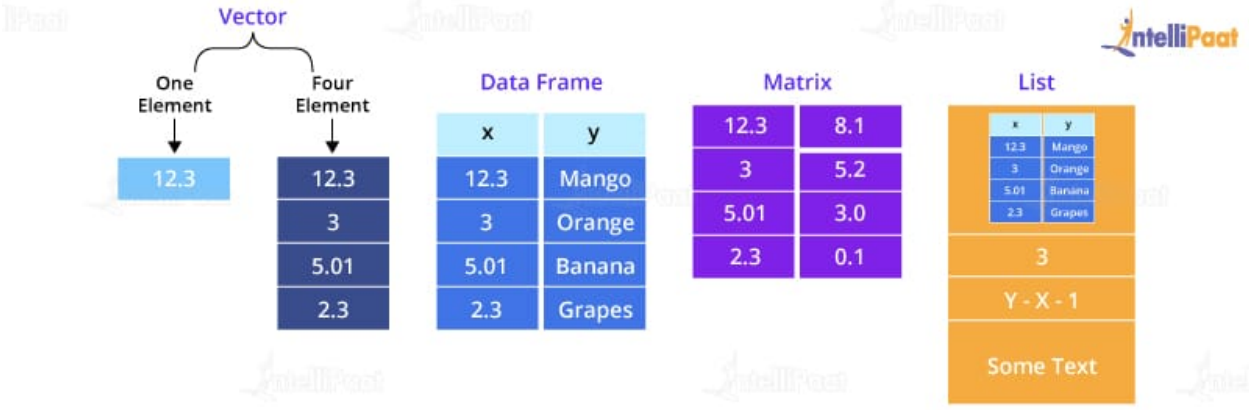
```r
0/0
```

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

```r
NA
```

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

2.2 常用数据结构



2.2.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

生成向量

- 通过 `c()` 来生成向量，只用按顺序输入向量的各个分量即可

```
v1 <- c(1,2,3,4)
v1
```

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

- 对于 $[1, 2, 3, 4, 5, ..., n]$ 这样的等差数列，可以使用 `1:n` 来直接生成

```
v2 <- 1:4
v2
```

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

- 对于有规则的等差数列，可以使用 `seq()` 来生成

```
seq(-2, 3, by=0.5) # 公差为 0.5, 首项为-2, 尾项最接近 3 的等差数列
```

```
##  [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
```

```
seq(2, 4, length.out=4) # 生成首项和尾项分别为 2 和 4 的 4 维向量
```

```
## [1] 2.000000 2.666667 3.333333 4.000000
```

```
seq(2, by=0.5, length=4)  # 生成首项为 2, 公差为 0.5, 长度为 4 的向量
```

```
## [1] 2.0 2.5 3.0 3.5
```

- **注意**：向量中的数据类型需要保持一致，否则会出现强制转换! 优先级为 字符型 > 数值型 > 逻辑型（优先级低的类型会被强制转换为优先级高的类型）

```
c(1,2,"3")    # 字符型最优先
```

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

```
c(1,2,TRUE)   # 其次是数值型
```

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

元素的索引和修改

- 使用 a[i] 来索引向量 a 中第 i 个元素

```
a <- c(1,2,3)
a[2]
```

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

- a[i] 中的 [i] 也可以是个列表

```
b <- c(2,2,2,1,3)
a[b]
```

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

```
c <- a >= 2    # 返回一个由逻辑值组成的向量
a[c]
```

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

- 通过对第 i 个元素进行赋值，来修改向量中第 i 个元素的值

```
a <- c(1,2,3)
a[1] <- 100
a
```

```
## [1] 100   2   3
```

```
a[2] = 300
a
```

```
## [1] 100 300   3
```

2.2.2 矩阵 (matrix)

- 为向量的推广，其元素具有**相同的数据类型**

矩阵的创建

- 通过函数 matrix(x,m,n) 的形式来生成矩阵（将向量 x 转换为 m 行 n 列的矩阵）
  - **列优先**

```
a <- 1:6
m1 <- matrix(a, 3, 2)
m1
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

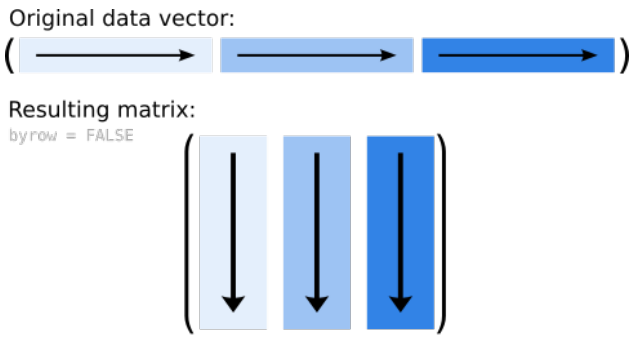


图 1: 默认为列优先

  - **行优先**

```
a <- 1:6
m2 <- matrix(a, 3, 2, byrow=T)
m2
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

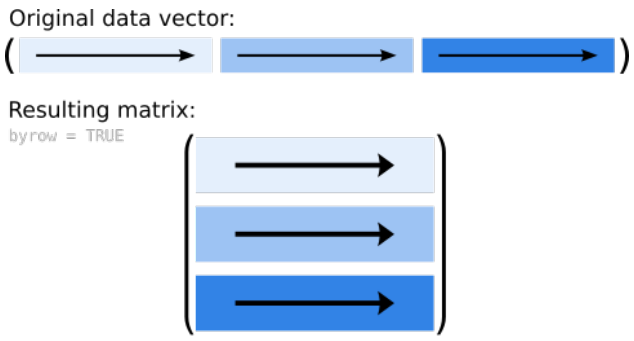


图 2: 通过 byrow=T 改为行优先

- **注意** 和向量一样，当矩阵中元素的类型不同时，也会强制转换元素类型：优先级也是 字符型 > 数值型 > 逻辑型（优先级低的类型会被强制转换为优先级高的类型）
  - 尝试如下代码:

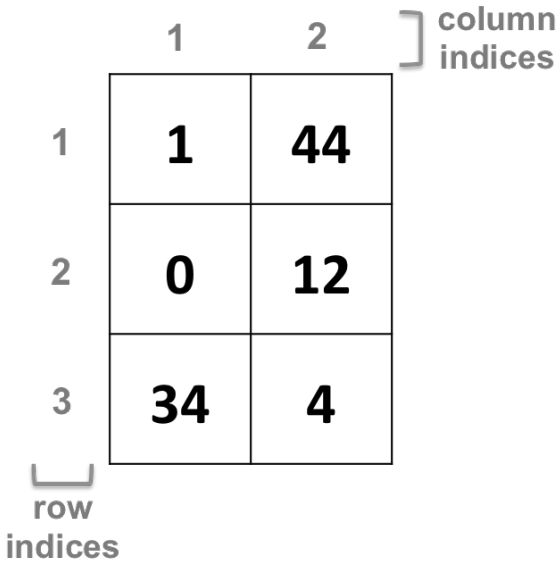
```
a1 <- c(1,2,3,4,"5",6)
m3 <- matrix(a1, 3,2)
m3



a2 <- c(T,2,3,4,5,6)
m4 <- matrix(a2, 3,2)
m4
```

矩阵的索引和元素的修改

- 和向量类似我们用 `a[i,j]` 的形式索引矩阵第 `i` 行 (row)，第 `j` 列 (column) 的元素;




```
x <- 1:9
m <- matrix(x, 3, 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
m[3,2] # 矩阵 m 第 3 行第 2 列的元素
```

```
## [1] 6
```

- 和向量类似，我们也可以用一组向量来索引矩阵

```
i <- c(1,3)
j <- c(1,2)
m[i,j] # 挑选出第 1 行和第 1,2 列，第 3 行和第 1,2 列相交的元素组成新的矩阵
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    3    6
```

- 索引某一列或某一行 `a[,n]` `a[m,]`

```
m[3,] # 索引第 3 行
```

```
## [1] 3 6 9
```

```
m[,2] # 索引第 2 列
```

```
## [1] 4 5 6
```

- 我们同样可以用赋值的方式来修改矩阵中元素的值

```
x <- 1:9
m <- matrix(x, 3, 3)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
i <- c(1,3)
j <- c(1,2)
m[i,j] <- 1:4  # 挑选出第 1 行和第 1,2 列，第 3 行和第 1,2 列相交的元素组成新的矩阵
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    7
## [2,]    2    5    8
## [3,]    2    4    9
```

2.2.3 数据框 (data frame)

- 属于矩阵的一种拓展形式，类似于我们常见的表格：每一行代表每一个 subject 的数据，每一列代表不同的数据（可能拥有不同的数据类型）

	x	y	sex
1	160	51	female
2	175	72	male

数据框的创建

- data.frame()

```
x <- c(160, 175)
y <- c(51, 72)
sex <- c("female", "male")
df <- data.frame(x,y,sex)                    # 使用列变量名 (x, y, sex)
df2 <- data.frame(height=x, weight=y, sex) # 设置列变量名分别为 (height, weight, sex)
```

数据框的索引

- names() 用于显示数据框各列的名称

```
names(df)
```

```
## [1] "x"   "y"   "sex"
```

```
names(df2)
```

```
## [1] "height" "weight" "sex"
```

- 对列进行索引：使用 df$ 列变量对于数据框的列进行索引，也可使用 df[j] 直接检索第 j 列

```
df$sex
```

```
## [1] "female" "male"
```

```
df[3]
```

```
##      sex
## 1 female
## 2   male
```

```
df2$height
```

```
## [1] 160 175
```

```
df2[2]
```

```
##   weight
## 1     51
## 2     72
```

- **对行进行索引**：使用 `df[i,]` 索引第 `i` 行 (**注意 `i` 后面有个 `,`**)

```
df[1,]
```

```
##     x  y    sex
## 1 160 51 female
```

```
df2[2,]
```

```
##   height weight  sex
## 2    175     72 male
```

- **对于元素进行索引**：使用 `df$x[i]` 索引第 `i` 个 subject 的 `x` 属性，类似于矩阵，我们也可以用行和列的方式进行索引 `df[i,j]` 索引第 `i` 行第 `j` 列的元素。

```
df$x[1]
```

```
## [1] 160
```

```
df[1,1]
```

```
## [1] 160
```

2.2.3 列表 (list)

- 列表可以视为向量在另一个方面的拓展，其内部可以存储不同类型的数据，且可根据键值 (key) 进行索引

列表的生成

- `list(key1=value1,key2=value2, ...)`

```
list1 <- list(1,"2",3)
list1
```

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

```
list2 <- list(x=1, y="2", z=3) # x,y,z 为我们索引使用的键值
list2
```

```
## $x
```

```
## [1] 1
##
## $y
## [1] "2"
##
## $z
## [1] 3
```

列表的索引

- 若我们像向量一样进行索引 `a[i]` 我们得到的还是一个列表而非列表内的值

```
class(list1[1]) # 运行后我们可以发现 list1[1] 返回的还是一个列表
```

```
## [1] "list"
```

- 我们需要使用 `a[[i]]` 才能得到具体的数值

```
class(list1[[1]]) # list[[1]] 返回的才是具体的元素值
```

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

- 类似于数据框，当我们定义键值后，可以使用 `list2$key` 来进行索引，这时可直接得到列表内元素的值

```
list2$y
```

```
## [1] "2"
```

```
class(list2$y) # $key 的方式可以直接返回列表的元素值
```

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

2.3 数值的基础运算和常用函数

- 基础运算：`+ - * / ^ sqrt()`

```
1 + 2 * 10    # 乘除以的优先级高于加减
```

```
## [1] 21
```

```
(1 + 2) * 10  # () 可以提高计算的优先级
```

```
## [1] 30
```

```
2 ^ 3     # 2 的 3 次方
```

```
## [1] 8
```

```r
2 ** 3    # 2 的 3 次方, 同上
```

```
## [1] 8
```

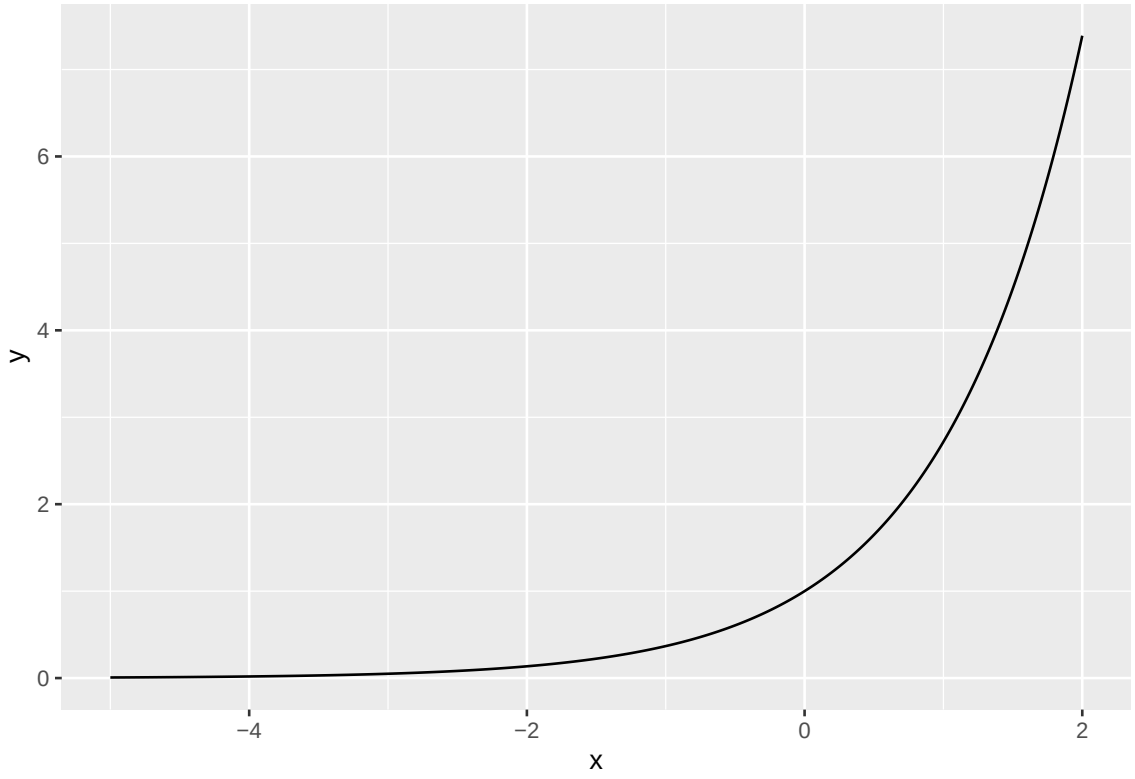
```r
sqrt(4)   # 对 4 开平方
```

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

- 常用函数

1. **指数函数**：`exp(x)` e^x ($e \approx 2.7182818284590452$ 称为自然常数)

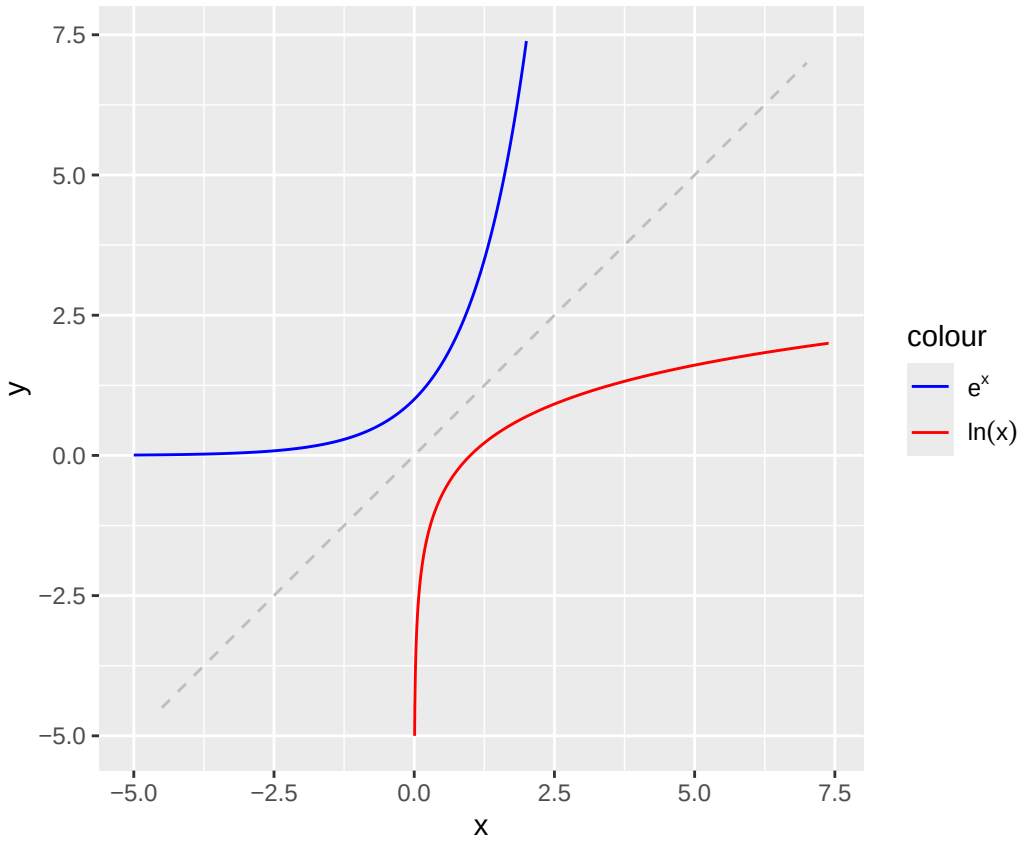
```r
x <- seq(-5,2,by=0.001)
y = exp(x)
```



2. **对数函数**：

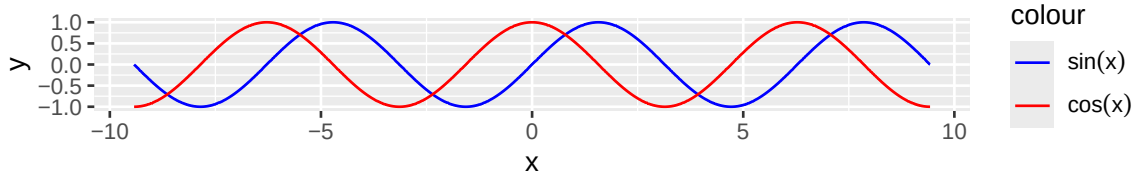
   - `log(x)` $\ln(x)$
   - `log(x, base=n)` $\log_n x$

```r
x <- seq(exp(-5),exp(2), by=0.001)
y = log(x)
```



3. 三角函数：sin(x) cos(x) 其中 x 采用弧度制

```
x <- seq(-3* pi,3*pi,by=1e-3)
y_sin = sin(x)
y_cos = cos(x)
```



4. sum(x) 求和函数：计算数值型向量 x 的所有元素的和；对于矩阵我们有 colSums() 对矩阵的列求和，rowSums() 对矩阵的行求和。

```
exp(1)
```

```
## [1] 2.718282
```

```
log(exp(1))
```

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

```
log(2^3, base=2) # 以 2 为底，8 的对数
```

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

```r
sin(pi/6)
```

```
## [1] 0.5
```

```r
x <- 1:9
sum(x)
```

```
## [1] 45
```

```r
m <- matrix(x,3,3)
colSums(m)
```

```
## [1]  6 15 24
```

```r
rowSums(m)
```

```
## [1] 12 15 18
```

Summary

1. 常见的变量类型：
    1. **逻辑型**：逻辑型的定义；逻辑型和数值型之间的转换；
    2. **数值型**：数值型的定义，科学计数法；
    3. **字符型**：字符型的定义；如何在字符串中引入 ' 或 "；（不同的引号，转义字符 \' \"
    4. **不同类型之间的转换**：逻辑型 → 字符型，数值型 → 字符型，字符型（部分）→ 数值型，逻辑型 ↔ 数值型；
    5. **特殊常量**：`pi` `Inf` `NaN` `NA`
2. 重要数据结构：
    1. **向量**：生成，索引，修改；强制转换
    2. **矩阵**：生成，索引，修改；强制转换
    3. **数据框**：生成，索引；
    4. **列表**：生成，索引；
3. 基本运算和函数