最基本的数据类型包括数字和字符串，科学计数法：1e-3 1e2，1L表示1这个整数

2 ^ 6 阶乘

5 %% 2 取余

数学函数log, log2, log10 exp(n)-e的n次方 sin, cos, tan sqrt

|或 &且 ≠0的数都是F

赋值：x<- 1 x = 1 1 -> x assign( "x", 1)

ls()显示当前环境下所有变量 rm( x )删除一个变量 rm(list=ls())删除当前环境下所有变量

特殊值：NULL：值未定义的表达式和函数；Inf：无限值（数字），分为+inf和-inf；NA：不可得到的值，不是有限值；NaN：不是数字（不存在）,属于NA的一种

c(-1, 0, 1) / 0 -Inf NaN Inf

判定特别值：is.finite()是不是有限值 is.infinite() is.na() is.nan() is.null()

判断数据类型：is.numeric() is.list()等，isTRUE函数只在参数是一个逻辑运算值true时才返回true，若传入的不是一个逻辑运算值，而是一个数字/字符等就都返回false

(x)中括号作用：如果不加括号，就不会输出y的值。括号将(x)整体作为一个表达式，这个表达式的值等于y的值，因此这一行实际上就是一个值，执行时会直接输出这个值

Vector类型转换优先级：逻辑类型->数字类型 逻辑类型->字符串 数字类型->字符串

矩阵内的数值必须是同一种类型，输入混合有多种基本数据类型时，矩阵会按上面提到的规则进行强制转换

ab <- c(a,b)合并ab数组 length()-长度 rev()-将数组倒序

矩阵matrix( c(20, 30.1, 2, 45.8, 23, 14), nrow = 2, byrow = T, dimnames = list( c("row\_A", "row\_B"), c("A", "B", "C") ) ) 矩阵的指定长度：nrow行 × ncol列。矩阵长度较小时，输入数据会被截短；而矩阵长度较大时，输入数据则会被重复使用，byrow-是否按行将数据转化为matrix，dimnames指定矩阵的行名和列名

vector和矩阵都是数组。vector是一维数组，矩阵是二维数组。还可以有更多维的数组，高维数组与vector和矩阵一样，只能包含一种基本数据类型。高维数组可以由函数 array() 定array(data=LETTERS[1:16],dim=c(2,4,2),dimnames=list(c("A","B"), c("one","two","three","four"), c("一", "二") ))

m[1:2, ] 按行数取行 把1:2放逗号后面就是取列，结果都是vector

m[c( "row\_B", "row\_A" ), ] 按行名取行

m[1:2, 2:3] 取其中一部分

m[ 1:2 , ] <- matrix( 1:6, nrow = 2 ) 替换

常用基础函数：

which()-遍历传入的vector，当vector中某元素满足判断条件时，which函数返回其索引

判断数据类型：str（）、class（）、typeof（）、mode();其中class、typeof显示整体数据类型，str显示变量内部每个元素类。mode函数只查看数据的大类，class函数查看数据的类，typeof函数则更细化，查看数据的细类

抽样：sample(x,size,replace = FALSE) x:数据size:抽样数replace:F，即抽样数size不能大于x长度

排序：sort()-输出排序后新序列 order()-输出新序列的每个元素在原序列中的位置

数据计算：range()-输出最大值和最小值 summary()-min,max,median,mean,1、3等分点 t()取二维矩阵的转置

改变数据类型：names( ab) <- as.character( ab) names把某个参数/行/列的名称改变，as.character将数字对象转为字符串类型

seq(0,10,along.with=c(1,10,2)), along.with=Id就是说seq生成数据序列的长度与Id序列长度相同，输出1 5 10

unique()-返回一个没有任何重复元素或行的向量、数据框或数组

subset(a,logical expression)根据逻辑表达式筛选a中的数据，返回令表达式为T的a中数据

vectorisation：R最重要的一个概念，核心在于数据自动循环使用

ab[3:6] <- c("2","3") 遍历赋值

c(1,2)\*c(2,3)就是c(1\*2,2\*3)

x<- c(4,6,5,7,10,9,4,15) ;x < c( 8, 4 )- x中的元素依次与8 4 8 4 8 4 8 4进行比较

data.frame与tibble: 二维表格，由不同列组成；每列是一个vector，不同列的数据类型可以不同，但一列只包括一种数据类型（int, num, chr …）各列的长度相同

data.frame( data = sample( 1:100, 10 ), group = sample( LETTERS[1:3], 10, replace = TRUE), data2 = 0.1 )创建3列，分别为data、group、data2

tribble(

~x, ~y, ~z, #~表示列名

"a", 2, 3.6,

"b", 1, 8.5)

tb<- tibble( x = character(), y = integer(), z = double() )规定这一列的类型

tb <- add\_row( .before=2)增加行, 在第二行之前插入 add\_column增加列

相互转换：as\_tibble()和as.data.frame

区别：

（1）tibble可以按顺序计算列值（将前一列的值用于创建下一列的值）tibble(x = 1:5, y = x ^ 2)，而data.frame不行

（2）tibble取一行/一列时(t[1,])返回的仍是一个tibble，而data.frame取一列(df[1,])时返回的是一个vector

（3）tibble可以进行可控的数据类型转换（使用[[]]或$时返回一个vector）

（4）tibble在根据列名取某列时不会部分匹配，更准确（data.frame(abc = 1)$ab不会报错，而tibble会报错）

（5）构造自动循环补全：tibble-仅限于长度为1或等长,而data.frame则为整除（data.frame(a = 1:6, b = LETTERS[1:2])可以），而tibble不行，只能tibble(a = 1, b = 1:3)

常用函数：

head(n=),tail(n=)显示前/后n行数据

nrow()行数 ncol()列数 dim()维度

str()表格结构：每列都是什么数据类型、有多少数据、数据值

rbind():根据行进行合并 cbind():增加列 如果一个df的行数少于另一处，就循环调用行数不足的vector

bind\_rows():合并tibble merge():默认按照相同的列名合并

tibble::enframe: 将vector或list转换为一列或两列的tibble。如enframe(c(1,2,3))，结果是两列的tibble，列名分别为name和value，列值都是(1,2,3)

res$Total=rowSums(res[,col\_list])求指定列的和，同理有rowMeans()

attach(iris):之后可以直接用sepal直接代指iris$sepal detach(iris)取消该操作，with(iris, head(Sepal.Length)也可

within(airquality,{rm(Day)}):删除airquality中Day列

readr包：read\_csv()逗号分隔符文件

read\_tsv(): tab separated files用tab分隔的文件

read\_delim(): general delimited files

read\_fwf(): fixed width files

read\_table(): tabular files where columns are separated by white-space.读取txt文件

read\_log(): web log files

使用：

read\_csv("data/talk03/iris.csv", col\_types = cols(Sepal.Length = col\_double(),Species = col\_character()))col\_character()指定读取的列类型, col\_names = FALSE不读取文件中的列名, col\_names = col\_name指定列名,n\_max = 7是从开始读取往后7行, quote = FALSE使输出的文本string两边没有引号, skip跳过前n行读取9

write\_tsv(iris, "iris.tsv", quote\_escape = "none")将iris写入"iris.tsv"中，同理有write\_csv、write\_delim、write\_excel\_csv、write\_file、write\_lines、write\_rds、write\_tsv

系统自带读取函数read.table、read.csv: header = TRUE显示列名, skip跳过前n行读取，quote消除string数据前后的字符, na.strings = NA让空缺数据显示为NA,col.names = col\_name指定列名,nrows = 7是从开始读取往后7行,sep=”;”让读取时按分号捕获数据

R session工作空间：save.image(file = "xx.RData")保存 load(file = "xx.RData")载入 save(city, country, file="1.RData")保存指定变量

向量：as.factor()，ordered = T参数时factors排序。Factors会限制输入数据的选择范围：新加入的值必须是levels里已存在的值，levels不可更改，若更改需调用函数levels(x) <- c(levels(x), "widowed”), levels的顺序决定了排序的顺序。用as.factor创建factor时，得到的levels按字母表排列；但是，用levels(y)方式指定levels时，则按照指定的顺序，sort函数也不会改变已指定的levels顺序

reorder(a,b,func)就是根据b对a:b这个键值对进行排序，排序方法为func（可为median等）

forcars包：

fct\_c：合并两个factor

fct\_inorder：按照第一次出现的次序排列

fct\_infreq：按照出现的频率排列（从大到小）

fct\_inseq：按照数字大小排列（从小到大）

fct\_count( gender );#统计gender里的levels数量

fct\_collapse(gender,Male=c("m", "male "))把gender中c("m", "male ")这些levels都改成Male，或使用fct\_relabel

#a %in% b 判断a是否在b中，若在就返回true