R Notebook

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

1. Interpreter Languag

R is a interpreter language. x <- 5 is "x=5 in c++"

```
# Comments
res <- (2+4)*5/2^-0.5
print(res)
```

```
## [1] 42.42641
```

2. R Object

R language is an object-oriented and function-oriented (Function directed) programming language.

```
int <- 5L
mode(int)
```

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

```
typeof(int)
```

```
## [1] "integer"
```

```
#vector
vec <- c(1,2)
mode(vec)
```

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

```
typeof(vec)
```

```
## [1] "double"
```

```
#logical
tf <- vec%%2 ==0
typeof(tf)
```

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

```
str <- c('A', 'B')
mode(str)
```

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

```
typeof(str)
```

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

```
#data frame
df <- data.frame(vec,str)
print(df)
```

```
##   vec str
## 1   1   A
## 2   2   B
```

```
typeof(df)
```

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

```
mode(df)
```

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

```
#list
lt <- list(vec,str)
print(lt)
```

```
## [[1]]
## [1] 1 2
##
## [[2]]
## [1] "A" "B"
```

```
mode(lt)
```

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

```
typeof(lt)
```

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

3. R Functions

log(x) log10(x) log(x,base)

max(x) min(x)

sum(x)

mean of x: mean(x)

variance of x: var(x)

standard deviation of x: sd(x)

```
log(100,10)
```

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

```
log(x=100, base=10)
```

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

4. R data structure

R language provides five main data structures: Vector, Matrix, Array, Data Frame, and Column List. These structures are specially designed to optimize data storage, access, processing or analysis. The data structure can be determined by its data dimension (1 dimensional, 2 dimensional or N-dimensional) and data type (homogeneous or heterogeneous) to distinguish.

Vector can be understood as a one-dimensional "array" (non-R language Array, similar to the array in C++ and Java), in R, create a The easiest way to measure is to use the c() function to combine the required data into A vector.

A vector can only store one type of data. If the type is different, the system will automatically Automatically converted to a compatible data type.

```
nums <- c(1, 2, 3)
print(nums)
```

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

```
multi <- c(1, 'B', TRUE)
print(multi)
```

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

```
print(nums[2])
```

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

```
# create 1-6 vector
seq_1 <- 1:6
print(seq_1)
```

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

```
# create 1-10 by 2

seq_2 <- seq(1, 10, by = 2)
print(seq_2)
```

```
## [1] 1 3 5 7 9
```

```
# create 1 1 1 1 1 1
rep_1 <- rep(1, 6)
print(rep_1)
```

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

```
length(seq_1)
```

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

Matrix can be understood as a two-dimensional array. In R, Use the matrix() function to create a matrix.

matrix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames=NULL)

```
# matrix(data=NA, nrow=1, ncol=1,
#        byrow=FALSE, dimnames=NULL)

m_1 <- matrix(1:12, 3, 4)
print(m_1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```r
data <- c('Leo', 'Male', '24',
          'Olivia', 'Female', '22')
rnames <- c('row1', 'row2')
cnames <- c('name', 'gender', 'age')

m_2 <- matrix(data = data,nrow = 2, ncol = 3,
              byrow = TRUE,
              dimnames = list(rnames, cnames))
print(m_2)
```

```
##      name     gender   age
## row1 "Leo"    "Male"   "24"
## row2 "Olivia" "Female" "22"
```

```r
# use the `dim()` function to get the number of rows and columns of the matrix:

dim(m_1)
```

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

Array can be understood as a multi-dimensional "array". Use the `array()` function to create an array in R. The `array()` function is defined as follows:

```r
# array(data=NA, dim=length(data), dimnames=NULL)

data <- c(25, 23, 30, 34, 18, 19, 20, 22)
dim_1 <- c('Day_1', 'Day_2')
dim_2 <- c('Loc_1', 'Loc_2', 'Loc_3', 'Loc_4')
arr_1 <- array(data, dim = c(2, 4), dimnames = list(dim_1, dim_2))
print(arr_1)
```

```
##       Loc_1 Loc_2 Loc_3 Loc_4
## Day_1    25    30    18    20
## Day_2    23    34    19    22
```

```r
data <- c('28', '90%', '102',
          '30', '80%', '100',
          '20', '69%', '90',
          '24', '86%', '97')
dim_1 <- c('Temperature',
           'Humidity', 'PM2.5')
dim_2 <- c('Day_1', 'Day_2')
dim_3 <- c('BJ', 'TJ')
arr <- array(data = data, dim = c(3, 2, 2),
             dimnames = list(dim_1, dim_2, dim_3))

print(arr)
```

```
## , , BJ
##
```

```
##             Day_1 Day_2
## Temperature "28"  "30"
## Humidity    "90%" "80%"
## PM2.5       "102" "100"
##
## , , TJ
##
##             Day_1 Day_2
## Temperature "20"  "24"
## Humidity    "69%" "86%"
## PM2.5       "90"  "97"
```

List can be understood as a collection of any objects in R. Unlike vectors, matrices and arrays, lists can store different types of data. In R, users can use `list()` to create lists, for example:

```r
lst <- list(1, c("Leo", "Tom"), max)
print(lst)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "Leo" "Tom"
##
## [[3]]
## function (..., na.rm = FALSE)  .Primitive("max")
```

A data frame can be understood as a collection of different data types (not arbitrary objects, excluding functions, etc.) in R. As one of the most commonly used data structures in R, use `data.frame()` to create a data frame in R, for example:

```r
id <- c(1, 2, 3)
name <- c('Leo', 'Tom', 'Olivia')
gender <- c('Male', 'Male', 'Female')
score <- c(90, 88, 96)
students <- data.frame(id, name, gender, score, stringsAsFactors = F)
print(students)
```

```
##   id   name gender score
## 1  1    Leo   Male    90
## 2  2    Tom   Male    88
## 3  3 Olivia Female    96
```

```r
print(students$name)
```

```
## [1] "Leo"    "Tom"    "Olivia"
```

Factors can be understood as the classification of other vector elements, used in R factor() creates a factor:

```r
stu_gender <- factor(c('Male', 'Female', 'Male', 'Male'))
print(stu_gender)
```

```
## [1] Male   Female Male   Male  
## Levels: Female Male
```

```r
stu_gender <- factor(c('Male', 'Female', 'Male', 'Male'), ordered = T)
print(stu_gender)
```

```
## [1] Male   Female Male   Male  
## Levels: Female < Male
```

```r
levels(stu_gender)
```

```
## [1] "Female" "Male"
```