

R 101

Shortcut Keys

- `ctrl + L` = clear หน้าจอ code
- `ctrl + enter` = run code

Function

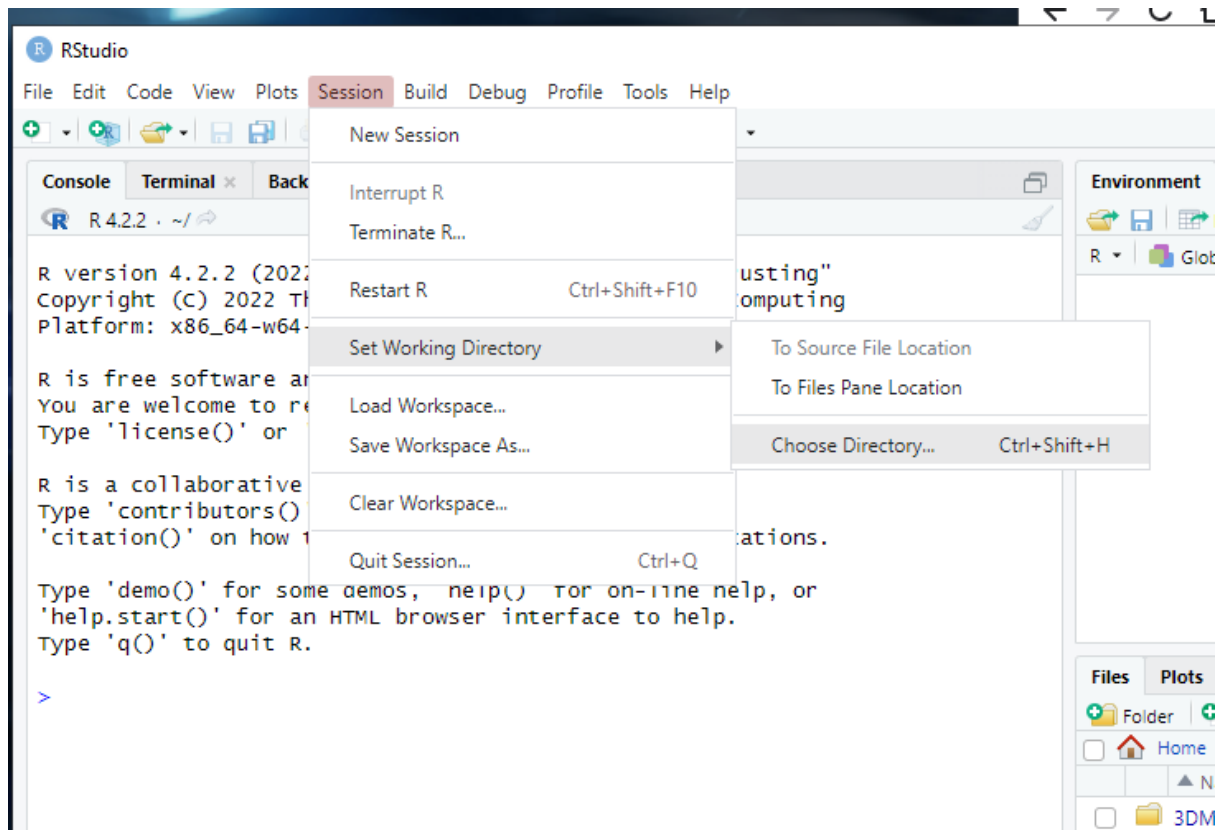
- `income` = สร้างตัวแปร
- `rm` = remove
- `==` = เปรียบเทียบ 2 ฝั่งมีค่าเท่ากัน
- `<=` = เปรียบเทียบ 2 ฝั่ง มีค่าน้อยกว่าหรือเท่ากับ
- `>=` = เปรียบเทียบ 2 ฝั่ง มีค่ามากกว่าหรือเท่ากับ
- `!=` = ไม่เท่ากับ
- `<` = น้อยกว่า
- `>` = มากกว่า
- `numeric` = ตัวเลขจำนวนเต็ม หรือทศนิยมก็ได้
- `logical` =
- `factor`
- `date`
- `C()` =
- `length` = ใช้ในการนับจำนวน Vector
- `dim` = ใส่ dimension ให้ Vector 5 แถว 5 column
- `NA` = ค่า Null
- `!` = เปลี่ยนค่าเป็นตรงข้ามเช่น เปลี่ยน true เป็น false

Set Working Directory

ก่อนเราเริ่มเขียน code ใน **RStudio** แอดแนะนำให้เราเช็ค directory ของเราก่อนนะครับ

[Working] **Directory** คือ folder ที่เราใช้ในการเก็บ code และ files ของโปรเจกต์เราทั้งหมด

ใน RStudio Cloud เราสามารถสร้าง folder ใหม่ แล้วก็เลือก folder นั้นเป็น working directory ได้เลย ไปที่เมนู **Session** > Set Working Directory > Choose Directory แล้วกดปุ่ม **New Folder** ตั้งชื่อโฟลเดอร์แล้วกด **Choose** เท่านั้นเสร็จเรียบร้อยแล้ว



ใช้ code `getwd()` เรียกดูที่เก็บไฟล์

```
getwd()
```

Create and Remove Variables

ใช้ `<-` ในการสร้าง

ใช้ `rm` ในการสร้าง remove ข้อมูล

สามารถ run code แค่นี้รอบได้

#ใช้ในการเขียนข้อความ

The screenshot shows the R Studio interface. The script editor on the left contains the following code:

```
1 ## create variables
2 income <- 50000
3 expense <- 30000
4 saving <- income - expense
5
6 ## remove variables
7 rm(saving)
8
```

The console on the right shows the execution of the code:

```
> income <- 50000
> 1
[1] 1
> income
[1] 50000
> expense <- 30000
> saving <- income - expense
> saving
[1] 20000
> rm(saving)
Warning message:
In rm(saving) : object 'saving' not found
> income <- 50000
> expense <- 30000
> saving <- income - expense
> |
```


The Environment pane at the bottom left shows the following values:

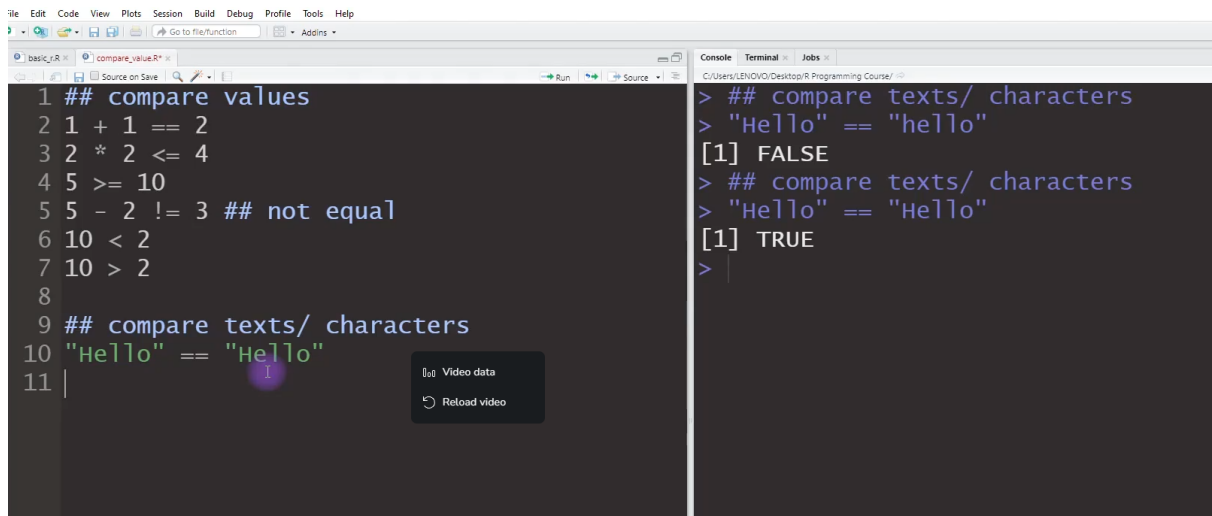
| Variable | Value |
|----------|-------|
| expense | 30000 |
| income | 50000 |
| saving | 20000 |

Comparison Operators

เราสามารถเขียน comparison operators ต่อไปนี้เพื่อเปรียบเทียบสองฝั่งของสมการได้ใน R

- `>`
- `>=`
- `<`
- `<=`
- `==` (equal)
- `!=` (not equal)

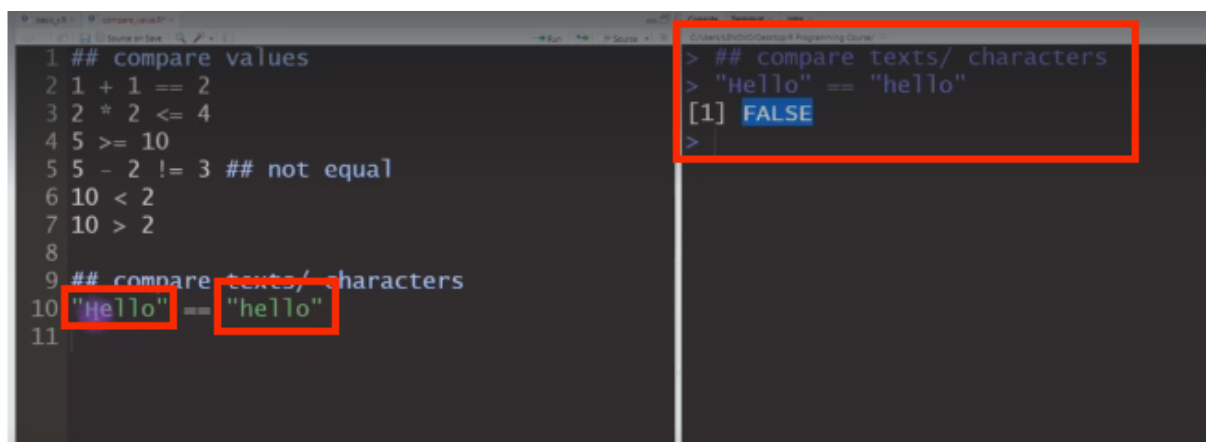
 Note - เราใช้เครื่องหมาย **double** equal signs `==` เพื่อเทียบสองฝั่งของสมการ ถ้าใช้ **single** equal sign เช่น `x = 5` จะเป็นการประกาศตัวแปร เพราะใน R เราสามารถสร้างตัวแปรได้สองแบบคือ `<-` หรือ `=`



```
1 ## compare values
2 1 + 1 == 2
3 2 * 2 <= 4
4 5 >= 10
5 5 - 2 != 3 ## not equal
6 10 < 2
7 10 > 2
8
9 ## compare texts/ characters
10 "Hello" == "Hello"
11 |
```

```
> ## compare texts/ characters
> "Hello" == "hello"
[1] FALSE
> ## compare texts/ characters
> "Hello" == "Hello"
[1] TRUE
> |
```

R เป็นภาษา Sensitive ตัวพิมพ์เล็ก-ใหญ่ มีผลทั้งหมด



```
1 ## compare values
2 1 + 1 == 2
3 2 * 2 <= 4
4 5 >= 10
5 5 - 2 != 3 ## not equal
6 10 < 2
7 10 > 2
8
9 ## compare texts/ characters
10 "Hello" == "hello"
11 |
```

```
> ## compare texts/ characters
> "Hello" == "hello"
[1] FALSE
> |
```

Data Types in R

ความรู้พื้นฐานเกี่ยวกับ data types คือสำคัญขั้นสุดตอนเขียนโปรแกรม ภาษาคอมพิวเตอร์
หลายๆภาษาจะ strict เรื่อง **data types** มากๆ ใน R ก็เช่นเดียวกัน โดย common data types
ที่ data analyst เราต้องใช้งานเป็นประจำจะมีอยู่ 5 ประเภทคือ

- `numeric`
- `character`
- `logical`
- `factor` อันนี้คือตัวแปร categorical ในทางสถิติ
- `date`

🌱 R มี package `lubridate` ที่ใช้จัดการ date ได้ง่ายมาก ไว้แอดมีสอนใน live class รอเรียน
ได้เลยครับ

```

## Data type

## 1.numeric
age <- 32
print(age)
class(age)

## 2. character
my_name <- "Toy"
my_university <- 'Bangkok University'
print(my_name)
print(my_university)
class(my_name); class(my_university)

## 3. logical
result <- 1+1 == 1
print(result)
class(result)

## 4.factor
animals <- c("Dog","Cat","Cat","Hippo")
class(animals)

factor <- factor(animals)
class(animals)

## 5. date/time
time_now <- Sys.time()
class(time_now)

```

Numeric ตัวเลขจำนวนเต็ม หรือทศนิยมก็ได้

The screenshot shows the R Studio environment. The editor window on the left contains the following code:

```

1 ## Data type
2
3 ## 1.numeric
4 age <- 32
5 print(age)
6 class(age)
7

```

The console window on the right shows the output of the code:

```

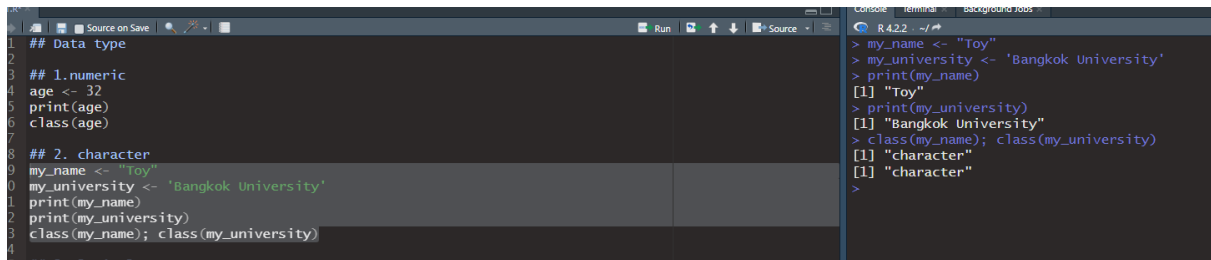
> ## 1.numeric
> age <- 32
> print(age)
[1] 32
> class(age)
[1] "numeric"
>

```

Character

การตั้งชื่อตัวแปร ชื่อจะอยู่ใน `" "` หรือ `' '` ก็ได้

หากต้องการ run code 2 อันสามารถใช้ตัว ; คั่นได้



```
## Data type
## 1.numeric
age <- 32
print(age)
class(age)

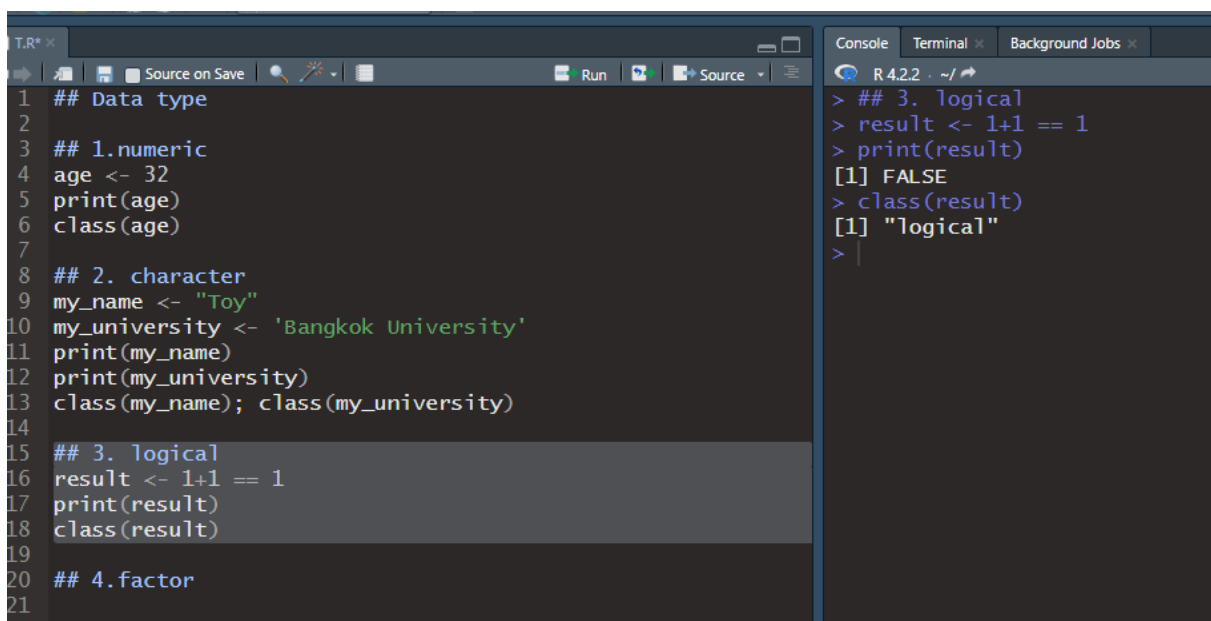
## 2. character
my_name <- "Toy"
my_university <- 'Bangkok University'
print(my_name)
print(my_university)
class(my_name); class(my_university)
```

Console

```
R 4.2.2 ~/\> my_name <- "Toy"
> my_university <- 'Bangkok University'
> print(my_name)
[1] "Toy"
> print(my_university)
[1] "Bangkok University"
> class(my_name); class(my_university)
[1] "character"
[1] "character"
>
```

Logical

Check Logi ถูกต้องหรือไม่ เป็น True หรือ False



```
## Data type
## 1.numeric
age <- 32
print(age)
class(age)

## 2. character
my_name <- "Toy"
my_university <- 'Bangkok University'
print(my_name)
print(my_university)
class(my_name); class(my_university)

## 3. logical
result <- 1+1 == 1
print(result)
class(result)

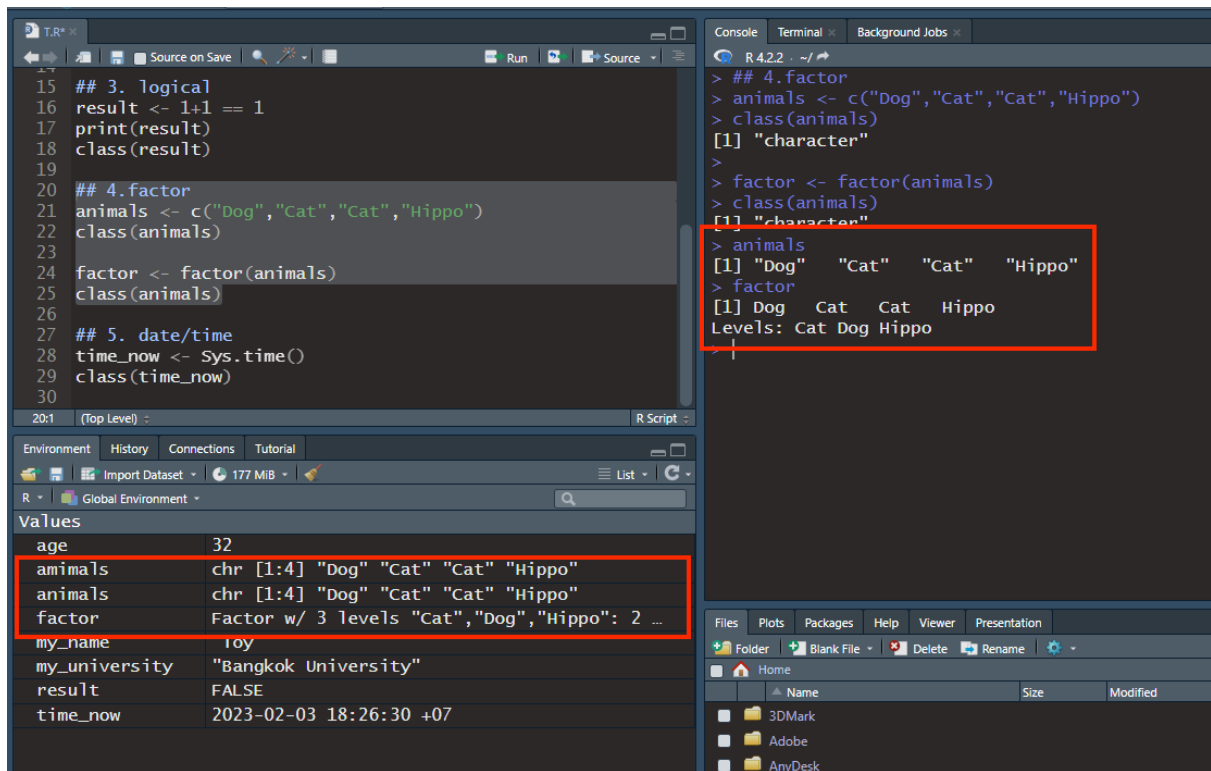
## 4.factor
```

Console

```
R 4.2.2 ~/\> ## 3. logical
> result <- 1+1 == 1
> print(result)
[1] FALSE
> class(result)
[1] "logical"
>
```

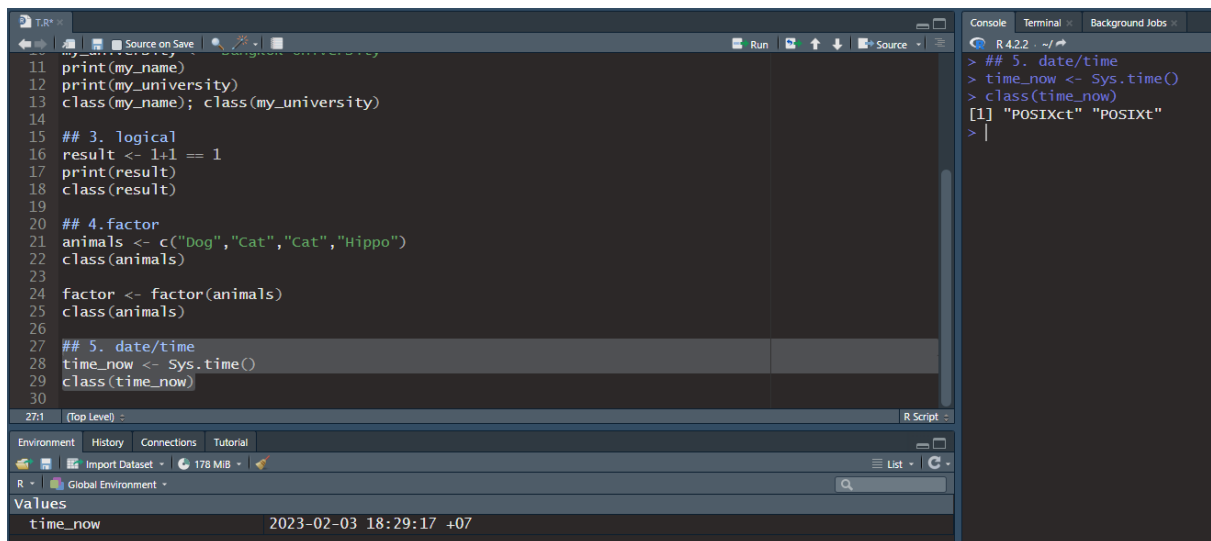
Factor

ใช้ในการจัดการตัวแปรกลุ่ม ยกตัวอย่างเช่น ผู้ชาย,ผู้หญิง



Date/Time

จะดึงข้อมูลจากวันที่และเวลาปัจจุบัน



Convert Data Types

การเปลี่ยน Type

```
## Convert Data Types

X <- 100
class(X)

char_x <- as.character(X)
num_x <- as.numeric(char_x)

## logical: TRUE/ FALSE
as.logical(0)
as.logical(1)
as.numeric(TRUE)
as.numeric(FALSE)
```

```
1 ## Convert Data Types
2
3 X <- 100
4 class(X)
5
6 char_x <- as.character(X)
7 num_x <- as.numeric(char_x)
8
9 ## logical: TRUE/ FALSE
10 as.logical(0)
11 as.logical(1)
12 as.numeric(TRUE)
13 as.numeric(FALSE)
14
```

```
> ## Convert Data Types
>
> X <- 100
> class(X)
[1] "numeric"
>
> char_x <- as.character(X)
> num_x <- as.numeric(char_x)
>
> ## logical: TRUE/ FALSE
> as.logical(0)
[1] FALSE
> as.logical(1)
[1] TRUE
> as.numeric(TRUE)
[1] 1
> as.numeric(FALSE)
[1] 0
>
```

| Values | |
|--------|-------|
| char_x | "100" |
| num_x | 100 |
| X | 100 |

Vector

```
## Data Structures
## 1. Vector
## 2. Matrix
```



```
## 3. List
## 4. DataFrame

## -----
## Vector

1:10
16:25

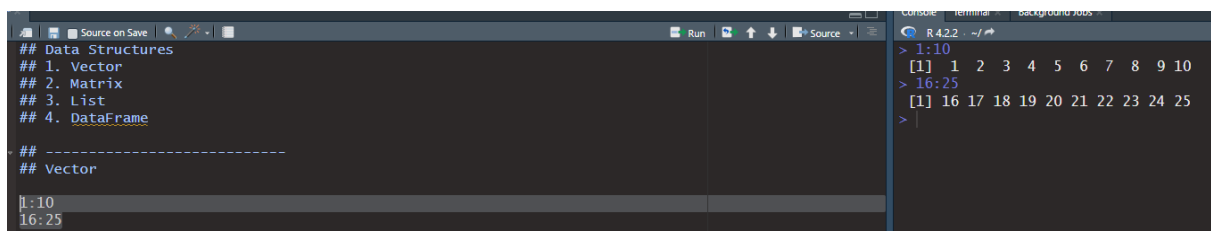
## sequence generation
seq(from = 1, to = 100, by = 5)

## help file
help("seq")

## function c
friend <- c("David", "Marry", "Anna", "John", "William")
ages <- c(30, 31, 25, 29, 32)
is_male <- c(TRUE, FALSE, FALSE, TRUE, TRUE)
```

Vector

สามารถใช้นับจำนวนได้โดยใช้ 



The screenshot shows the R Studio interface. The source editor on the left contains the following code:

```
## Data Structures
## 1. Vector
## 2. Matrix
## 3. List
## 4. DataFrame

## -----
## Vector

1:10
16:25
```

The console on the right shows the execution results:

```
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> 16:25
[1] 16 17 18 19 20 21 22 23 24 25
```

Query ข้อมูล 1-100 เพื่อนไข่มุ่บับตัวที่ 2

```
## Data Structures
## 1. Vector
## 2. Matrix
## 3. List
## 4. Data-frame

## -----
## Vector

1:10
16:25

## sequence generation
seq(from = 1, to = 100, by = 5)
```

```
> ## sequence generation
> seq(from = 1, to = 100, by = 5)
[1] 1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
> |
```

```
seq(from = 1, to = 100, by = 5)
```

ใช้ในการช่วยหา Function

```
> ## help file
> help("seq")
> |
```

R: Sequence Generation - Find in Topic

type "integer" or "double": programmers should not rely on which.

`seq_along` and `seq_len` return an integer vector, unless it is a *long vector* when it will be double.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

The methods `seq.Date` and `seq.POSIXt`.

[i](#), [rep](#), [sequence](#), [row.col](#).

Examples

[Run examples](#)

```
seq(0, 1, length.out = 11)
seq(stats::rnorm(20)) # effectively 'along'
seq(1, 9, by = 2)      # matches 'end'
seq(1, 9, by = pi)     # stays below 'end'
seq(1, 6, by = 3)
seq(1.575, 5.125, by = 0.05)
seq(17) # same as 1:17, or even better seq_len(17)
```

[Package base version 4.2.2 [Index](#)]

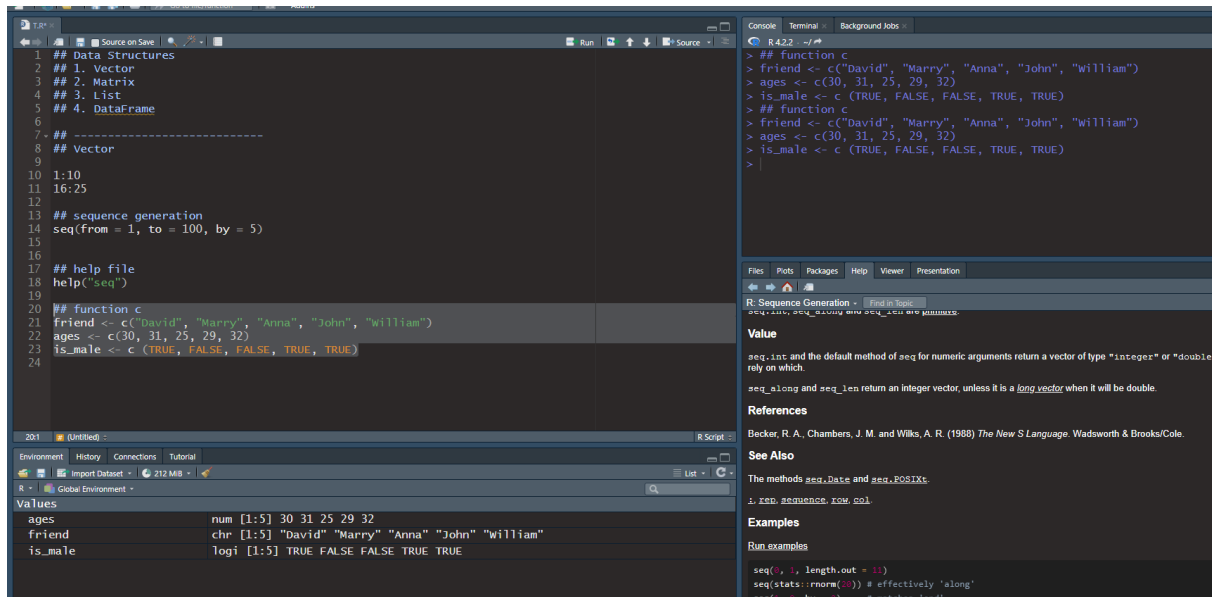
```
## help file
```

```
help("seq")
```

Function Vector C

Vector เก็บข้อมูลได้แค่ 1 ประเภทใน ()

ใช้ Function C เพื่อช่วยในการกำหนดประเภท



The screenshot shows the RStudio environment. The script editor on the left contains the following code:

```
## Data Structures
## 1. Vector
## 2. Matrix
## 3. List
## 4. DataFrame
#####
## Vector
1:10
16:25
## sequence generation
seq(from = 1, to = 100, by = 5)
## help file
help("seq")
## function c
friend <- c("David", "Marry", "Anna", "John", "William")
ages <- c(30, 31, 25, 29, 32)
is_male <- c(TRUE, FALSE, FALSE, TRUE, TRUE)
```

The console on the right shows the execution of the code, and the environment pane at the bottom displays the variables created:

| Variable | Value |
|----------|---|
| ages | num [1:5] 30 31 25 29 32 |
| friend | chr [1:5] "David" "Marry" "Anna" "John" "William" |
| is_male | logi [1:5] TRUE FALSE FALSE TRUE TRUE |

The right pane shows the help documentation for the `seq` function, including its value, references, and examples.

Matrix

เก็บข้อมูลได้ 1 ประเภทเหมือน Vector

Example

x สร้างเลข 1-25

length = ใช้ในการนับจำนวน Vector

dim = ใส่ dimension ให้ Vector 5 แถว 5 column

สามารถใช้ Function matrix ในการสร้างได้ ได้ผลลัพธ์เหมือน dim

The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
1 ## Matrix
2 x <- 1:25
3 length(x)
4 dim(x) <- c(5,5)
5
6 m1 <- matrix(1:25, ncol=5)
7 m2 <- matrix(1:6, ncol=3, nrow=2, byrow=T)
```

The Environment pane shows the objects: `m1` (int [1:5, 1:5]), `m2` (int [1:2, 1:3]), and `x` (int [1:5, 1:5]). The console on the right shows the output of the code:

```
> x <- 1:25
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
[16] 16 17 18 19 20 21 22 23 24 25
> length(x)
[1] 25
> dim(x) <- c(5,5)
> x
      [,1] [,2] [,3] [,4] [,5]
[1,]  1    6   11   16   21
[2,]  2    7   12   17   22
[3,]  3    8   13   18   23
[4,]  4    9   14   19   24
[5,]  5   10   15   20   25
> m1 <- matrix(1:25, ncol=5)
> m1
      [,1] [,2] [,3] [,4] [,5]
[1,]  1    6   11   16   21
[2,]  2    7   12   17   22
[3,]  3    8   13   18   23
[4,]  4    9   14   19   24
[5,]  5   10   15   20   25
> m2 <- matrix(1:6, ncol=3, nrow=2, byrow=T)
> m2
      [,1] [,2] [,3]
[1,]  1    2    3
[2,]  4    5    6
```

```
## Matrix
x <- 1:25
length(x)
dim(x) <- c(5,5)

m1 <- matrix(1:25, ncol=5)
m2 <- matrix(1:6, ncol=3, nrow=2, byrow=T)
```

This screenshot is similar to the first one, but with a red rectangular box highlighting the console output for the `m1` and `m2` matrices. The script editor shows the same code as before. The console output is identical to the first screenshot, but the portion from `> m1` to `> m2` is enclosed in a red box.

m1 = ใช้ ncol เป็นการสร้างหัวข้อ 5 column แล้วนับเลข 1-2

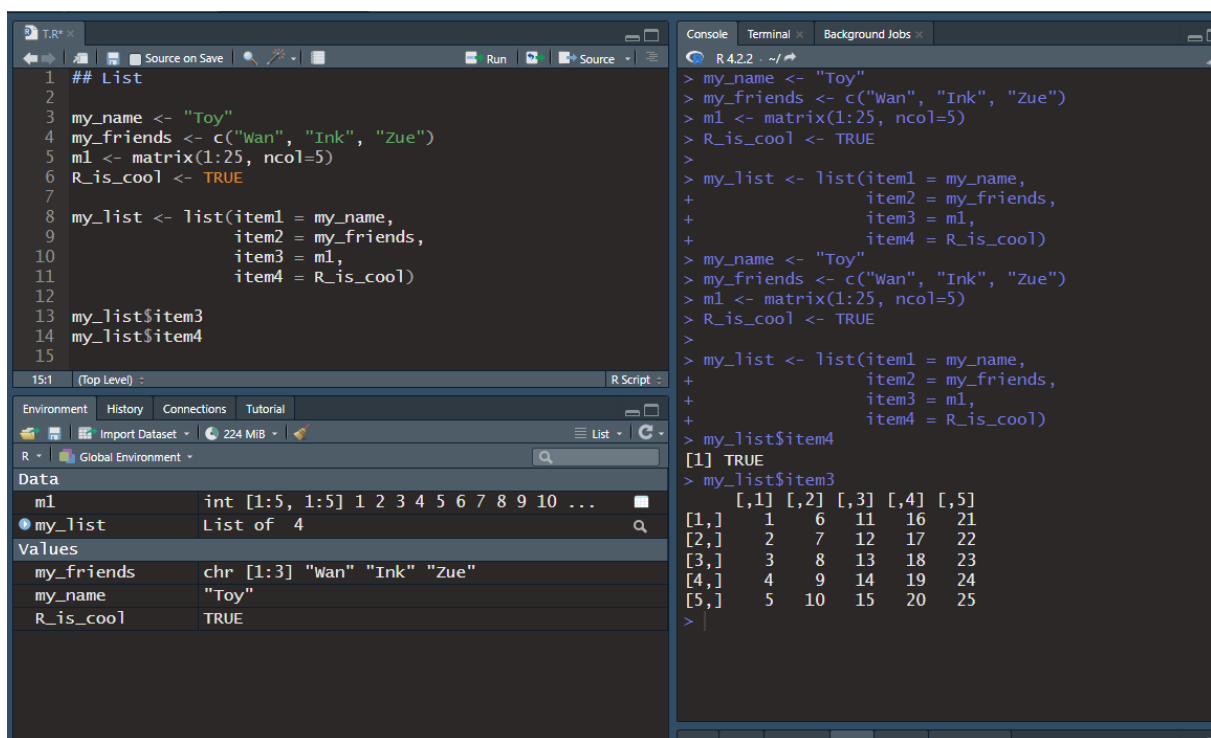
m2 = ใช้ ncol สร้าง 3 column 2 row นับเลข 1-6

สร้างเขียนให้บวกได้

```
[2,] 4 5 6
> m2 + 100
      [,1] [,2] [,3]
[1,] 101 102 103
[2,] 104 105 106
>
```

List

สามารถสร้าง list และเขียน \$ ในการเลือกดึงข้อมูลได้เร็วขึ้น



The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
## List
1
2
3 my_name <- "Toy"
4 my_friends <- c("Wan", "Ink", "Zue")
5 m1 <- matrix(1:25, ncol=5)
6 R_is_cool <- TRUE
7
8 my_list <- list(item1 = my_name,
9                 item2 = my_friends,
10                item3 = m1,
11                item4 = R_is_cool)
12
13 my_list$item3
14 my_list$item4
15
```

The console on the right shows the execution of the code:

```
> my_name <- "Toy"
> my_friends <- c("Wan", "Ink", "Zue")
> m1 <- matrix(1:25, ncol=5)
> R_is_cool <- TRUE
>
> my_list <- list(item1 = my_name,
+                 item2 = my_friends,
+                 item3 = m1,
+                 item4 = R_is_cool)
> my_name <- "Toy"
> my_friends <- c("Wan", "Ink", "Zue")
> m1 <- matrix(1:25, ncol=5)
> R_is_cool <- TRUE
>
> my_list <- list(item1 = my_name,
+                 item2 = my_friends,
+                 item3 = m1,
+                 item4 = R_is_cool)
> my_list$item4
[1] TRUE
> my_list$item3
      [,1] [,2] [,3] [,4] [,5]
[1,] 1 6 11 16 21
[2,] 2 7 12 17 22
[3,] 3 8 13 18 23
[4,] 4 9 14 19 24
[5,] 5 10 15 20 25
>
```

The Environment pane at the bottom shows the objects created:

- m1: int [1:5, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
- my_list: List of 4
- my_friends: chr [1:3] "Wan" "Ink" "Zue"
- my_name: "Toy"
- R_is_cool: TRUE

```
## List
```

```
my_name <- "Toy"
my_friends <- c("Wan", "Ink", "Zue")
m1 <- matrix(1:25, ncol=5)
R_is_cool <- TRUE
```

```
my_list <- list(item1 = my_name,  
               item2 = my_friends,  
               item3 = m1,  
               item4 = R_is_cool)  
  
my_list$item3  
my_list$item4
```

Data Frame

สามารถใช้ Function Data Frame เพื่อสร้างข้อมูล Table ได้

```
## Data Frame  
  
friends <- c ("Wan", "Ink", "Aan",  
             "Bee", "Top")  
  
ages <- c (26, 27 ,32, 31 ,28)  
  
locations <- c ("New York", "London",  
               "London", "Tokyo",  
               "Manchester")  
  
movie_lover <- c (TRUE, TRUE, FALSE,  
                 TRUE, TRUE)  
  
df <- data.frame(friends,  
                 ages,  
                 locations,  
                 movie_lover)  
  
View(df)
```

The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
1 ## Data Frame
2
3 friends <- c ("Wan", "Ink", "Aan",
4             "Bee", "Top")
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9               "London", "Tokyo",
10              "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                 TRUE, TRUE)
14
15 data.frame(friends,
16           ages,
17           locations,
18           movie_lover)
19
```

The console on the right shows the execution of the code, resulting in the following data frame:

```
> ## Data Frame
>
> friends <- c ("Wan", "Ink", "Aan",
+             "Bee", "Top")
>
> ages <- c (26, 27 ,32, 31 ,28)
>
> locations <- c ("New York", "London",
+               "London", "Tokyo",
+               "Manchester")
>
> movie_lover <- c (TRUE, TRUE, FALSE,
+                 TRUE, TRUE)
>
> data.frame(friends,
+           ages,
+           locations,
+           movie_lover)
  friends ages locations movie_lover
1   Wan    26  New York      TRUE
2   Ink    27   London      TRUE
3   Aan    32   London     FALSE
4   Bee    31   Tokyo       TRUE
5   Top    28 Manchester      TRUE
```

The Environment pane at the bottom shows the data frame 'm1' as an integer matrix with dimensions 5x4.

ย่อ data.frame เป็น df ได้โดยเขียน df <-

The screenshot shows the RStudio script editor with the following code:

```
df <- data.frame(friends,
                  ages,
                  locations,
                  movie_lover)

view(df)
```

ใช้ View ในการดูแบบ Table

| | friends | ages | locations | movie_lover |
|---|---------|------|------------|-------------|
| 1 | Wan | 26 | New York | TRUE |
| 2 | Ink | 27 | London | TRUE |
| 3 | Aan | 32 | London | FALSE |
| 4 | Bee | 31 | Tokyo | TRUE |
| 5 | Top | 28 | Manchester | TRUE |

สามารถเปลี่ยนชื่อ Column โดยใช้ list เข้าไป

```

2 friends <- c ("Wan", "Ink", "Aan",
3             "Bee", "Top")
4
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9               "London", "Tokyo",
10              "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                 TRUE, TRUE)
14
15 ## create dataframe from list
16 my_list <- list(friends = friends,
17               ages = ages,
18               locations = locations,
19               movie = movie_lover)
20
21 data.frame(my_list)

```

```

> View(df)
> friends <- c ("Wan", "Ink", "Aan",
+             "Bee", "Top")
> ## Data Frame
>
> friends <- c ("Wan", "Ink", "Aan",
+             "Bee", "Top")
>
> ages <- c (26, 27 ,32, 31 ,28)
>
> locations <- c ("New York", "London",
+               "London", "Tokyo",
+               "Manchester")
>
> movie_lover <- c (TRUE, TRUE, FALSE,
+                 TRUE, TRUE)
>
> ## create dataframe from list
> my_list <- list(friends = friends,
+               ages = ages,
+               locations = locations,
+               movie = movie_lover)
>
> data.frame(my_list)
  friends ages locations movie
1   Wan    26  New York  TRUE
2   Ink    27   London  TRUE
3   Aan    32   London FALSE
4   Bee    31   Tokyo   TRUE
5   Top    28 Manchester  TRUE

```

Environment: History Connections Tutorial

Global Environment

Data

my_list List of 4

Values

ages num [1:5] 26 27 32 31 28

friends chr [1:5] "Wan" "Ink" "Aan" "Bee" "Top"

locations chr [1:5] "New York" "London" "London" "To..."

movie_lover logi [1:5] TRUE TRUE FALSE TRUE TRUE

R: Sequence Generation - Find in Topic

type "integer" or "double": programmers should not rely on which.

Data Frame

```
friends <- c ("Wan", "Ink", "Aan",
             "Bee", "Top")
```

```
ages <- c (26, 27 ,32, 31 ,28)
```

```
locations <- c ("New York", "London",
```



```

      "London", "Tokyo",
      "Manchester")

movie_lover <- c (TRUE, TRUE, FALSE,
                 TRUE, TRUE)

## create dataframe from list
my_list <- list(friends = friends,
               ages = ages,
               locations = locations,
               movie = movie_lover)

data.frame(my_list)

```

ถ้าต้องการดึงข้อมูลทุกแถว และคอลัมน์ 2,4,5 ของ dataframe customers ต้องเขียน [] อย่างไร? (เลือกสองคำตอบ)

customers[1:100 , (2,4,5)]

customers[: , c(2,4,5)]

customers[: , (2,4,5)]

correct

customers[, c(2,4,5)]



correct

customers[c(2,4,5)]



< Back

Continue >

Subset

ใช้ `[]` ในการดึงข้อมูลจาก Table ที่สร้าง สามารถสลับกันได้

```

1 ## Data Frame
2
3 friends <- c ("Wan", "Ink", "Aan",
4             "Bee", "Top")
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9               "London", "Tokyo",
10              "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                 TRUE, TRUE)
14
15 df <- data.frame(friends,
16                 ages,
17                 locations,
18                 movie_lover)
19
20 view(df)

```

```

> friends[1]
[1] "Wan"
> friends[2]
[1] "Ink"
> friends[5]
[1] "Top"
> friends[1:3]
[1] "Wan" "Ink" "Aan"
> friends[4:5]
[1] "Bee" "Top"
> friends[ c(1,3,5)]
[1] "Wan" "Aan" "Top"
>

```

สามารถดึงมากกว่าหรือน้อยกว่าได้

```

1 ## Data Frame
2
3 friends <- c ("Wan", "Ink", "Aan",
4             "Bee", "Top")
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9               "London", "Tokyo",
10              "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                 TRUE, TRUE)
14
15 df <- data.frame(friends,
16                 ages,
17                 locations,
18                 movie_lover)
19
20 view(df)

```

```

> ages
[1] 26 27 32 31 28
> ages[ ages > 30]
[1] 32 31
> ages[ ages <= 30]
[1] 26 27 28
>

```

สามารถนำชื่อเพื่อนเข้าไปใน ages ได้

```
1 ## Data Frame
2
3 friends <- c ("Wan", "Ink", "Aan",
4              "Bee", "Top")
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9                "London", "Tokyo",
10               "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                  TRUE, TRUE)
14
15 df <- data.frame(friends,
16                 ages,
17                 locations,
18                 movie_lover)
19
20 view(df)
21 |
```

```
> ages
[1] 26 27 32 31 28
>
> names(ages) <- friends
>
> ages
Wan Ink Aan Bee Top
26 27 32 31 28
> |
```

ดึงข้อมูลชื่อและอายุแบบสลับตำแหน่งได้

```
1 ## Data Frame
2
3 friends <- c ("Wan", "Ink", "Aan",
4              "Bee", "Top")
5
6 ages <- c (26, 27 ,32, 31 ,28)
7
8 locations <- c ("New York", "London",
9                "London", "Tokyo",
10               "Manchester")
11
12 movie_lover <- c (TRUE, TRUE, FALSE,
13                  TRUE, TRUE)
14
15 df <- data.frame(friends,
16                 ages,
17                 locations,
18                 movie_lover)
19
20 View(df)
21
22
```

```
> ages
Wan Ink Aan Bee Top
26 27 32 31 28
>
> ages["Wan"]
Wan
26
>
> ages["Top"]
Top
28
>
> ages[ c("Wan", "Aan", "Bee")]
Wan Aan Bee
26 32 31
>
> ages[ c("Bee", "Wan", "Aan")]
Bee Wan Aan
31 26 32
>
```

การดึงข้อมูลจาก Data frame [row, column]

The screenshot shows the RStudio environment with a data frame 'df' and its console output. The data frame 'df' has 5 rows and 4 columns: friends, ages, locations, and movie_lover. The console shows the following commands and their outputs:

```

> df[1, 3]
[1] "New York"
> df[2, 4]
[1] TRUE
> df[1:2, 4]
[1] TRUE TRUE
> df[1:2, 2:4]
  ages locations movie_lover
1  26 New York      TRUE
2  27  London      TRUE
> df[, "friends"]
[1] "Wan" "Ink" "Aan" "Bee" "Top"
> df[, c("friends", "locations")]
  friends locations
1   Wan   New York
2   Ink    London
3   Aan    London
4   Bee   Tokyo
5   Top  Manchester

```

สามารถใช้ df ดึงข้อมูลภายใต้เงื่อนไขได้

- 1.ดึงข้อมูลเฉพาะ movie = TRUE
- 2.ดึงข้อมูลเฉพาะ movie = FALSE
- 3.ดึงข้อมูล ages ที่อายุน้อยกว่า 30
- 4.ดึงข้อมูลคนชื่อ Top ขึ้นมา



สังเกตด้านหลังจะเป็นเครื่องหมายวงเล็บว่างไว้ เป็นเงื่อนไขให้ดึงทุก Column

The screenshot shows the RStudio environment with a data frame 'df' and its console output. The data frame 'df' has the following structure:

| | friends | ages | locations | movie_lover |
|---|---------|------|------------|-------------|
| 1 | Wan | 26 | New York | TRUE |
| 2 | Ink | 27 | London | TRUE |
| 3 | Aan | 32 | London | FALSE |
| 4 | Bee | 31 | Tokyo | TRUE |
| 5 | Top | 28 | Manchester | TRUE |

The console output shows the following R commands and their results:

```

> df[ df$movie_lover == TRUE, ]
  friends ages locations movie_lover
1   Wan   26   New York         TRUE
2   Ink   27    London         TRUE
4   Bee   31    Tokyo         TRUE
5   Top   28 Manchester         TRUE

> df[ df$movie_lover == FALSE, ]
  friends ages locations movie_lover
3   Aan   32    London         FALSE

> df[ df$ages < 30, ]
  friends ages locations movie_lover
1   Wan   26   New York         TRUE
2   Ink   27    London         TRUE
5   Top   28 Manchester         TRUE

> df[ df$friends == "Top", ]
  friends ages locations movie_lover
5   Top   28 Manchester         TRUE

```

```
## 1
> df[ df$movie_lover == TRUE, ]
  friends ages locations movie_lover
1   Wan   26   New York         TRUE
2   Ink   27    London         TRUE
4   Bee   31    Tokyo         TRUE
5   Top   28 Manchester         TRUE
```

```
## 2
> df[ df$movie_lover == FALSE, ]
  friends ages locations movie_lover
3   Aan   32    London         FALSE
```

```
## 3
> df[ df$ages < 30, ]
  friends ages locations movie_lover
1   Wan   26   New York         TRUE
2   Ink   27    London         TRUE
5   Top   28 Manchester         TRUE
```

```
## 4
> df[ df$friends == "Top", ]
```

```
      friends ages  locations movie_lover  
5      Top    28 Manchester      TRUE  
>
```