

R 101

| | |
|-----------------|---|
| ☰ Tags | Programming R |
| ➤ Class | |
| ☑ Finished Yet? | ☑ |
| ➤ Knowledge |  The Third Sprint: R |

Meet RStudio Cloud

-เราสามารถเขียน R ผ่าน Web Browser ได้ฟรี 25 ชั่วโมงต่อเดือนที่ RStudio Cloud: <https://posit.cloud/>

-ถ้าไม่อยากจะเขียนผ่าน Web Browser สามารถโหลด Base R (<https://cran.r-project.org/bin/windows/base/>) และ RStudio (<https://posit.co/download/rstudio-desktop/>) ลงเครื่องได้เช่นกัน

[ลง Base R ก่อนแล้วค่อยลง RStudio ตาม]

Lesson 1: Download and Install Software

-เราจะต้อง Install 2 ส่วน คือ Base R (Core) และ RStudio Desktop (IDE)

-ตอนโหลด RStudio Desktop ให้เลือกตัวฟรี

How to use setwd() on RStudio Cloud

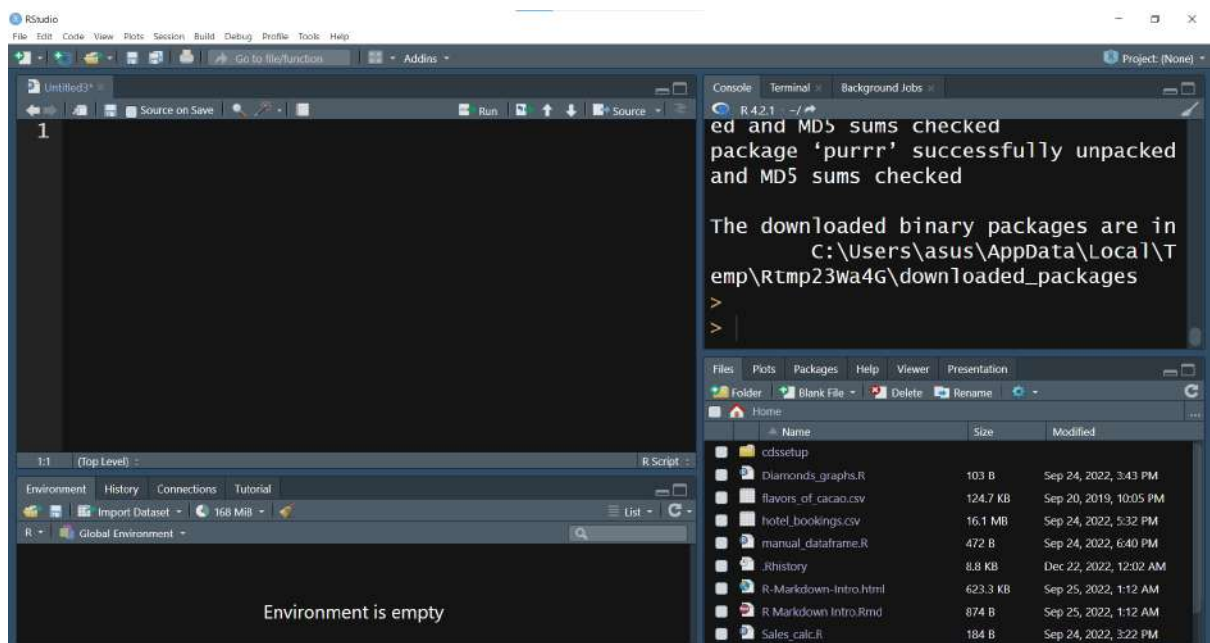
-ก่อนเริ่มเขียนโค้ด ให้เช็ค directory ก่อน

-[Working] Directory คือ Folder ที่ใช้ในการเก็บไฟล์การทำงานของเรทั้งหมด ใน RStudio Cloud เราสามารถสร้าง Folder ใหม่ แล้วเลือก Folder นั้นเป็น Working Directory ได้เลย [Session → Set Working Directory → Choose Directory] แล้วกดปุ่ม New Folder → ตั้งชื่อ Folder → กด Choose

Lesson 2: Create and Remove Variable

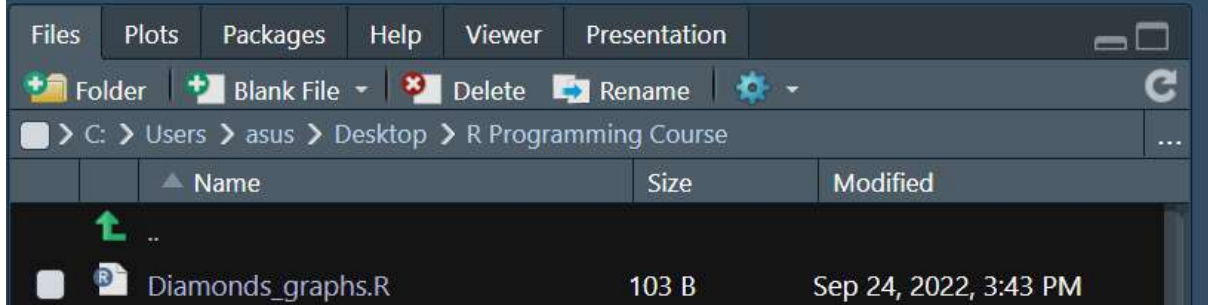
-เราสามารถปรับ Font และขนาดของตัวอักษรได้ที่ Tools → Global Options (ล่างสุด) → Appearance (นอกจากนี้ เราสามารถปรับ Theme ให้พื้นหลังมืดลงเพื่อความสบายตาได้ด้วย)

-เราสามารถปรับ Pane Layout ให้ Console อยู่ข้างขวาแทนข้างล่างได้ เช่น:

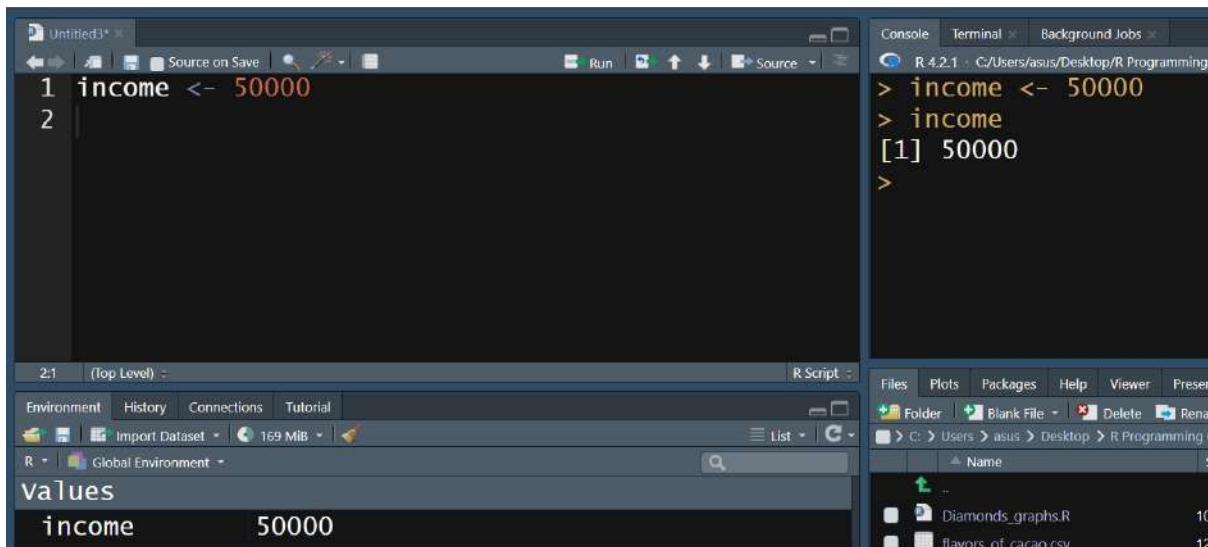


-เราสามารถใช้ getwd() เพื่อเช็ค Working Directory ที่เก็บไฟล์ของเราได้ หรือจะดูที่แถบข้างบนก็ลองก็ได้เช่นกัน เช่น:

```
> getwd()
[1] "C:/Users/asus/Desktop/R Programm
ing Course"
> |
```

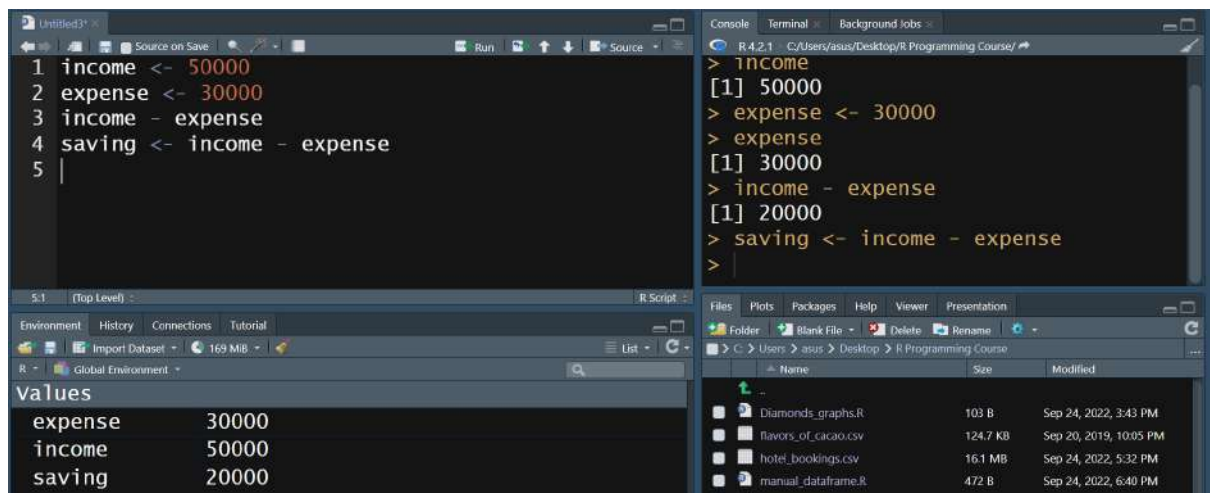


- เราสามารถกด CTRL+L เพื่อล้าง Console ได้
- หลังจากที่เขียนโค้ดเสร็จ ให้กด CTRL+Enter เพื่อรันโค้ด
- วิธีการสร้างตัวแปร:



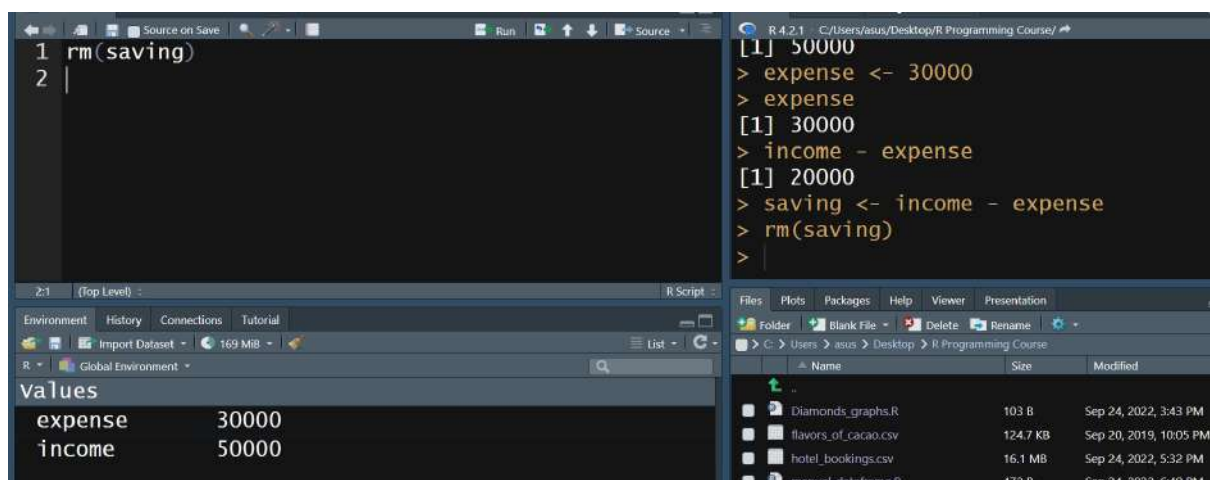
[ชื่อตัวแปร ← ค่าของตัวแปร เราจะใช้ลูกศรในการ Assign ค่ากับตัวแปร เมื่อเรากำหนดค่าของตัวแปรแล้วพิมพ์ชื่อตัวแปรใน Console ระบบจะแสดงค่าของตัวแปรขึ้นมาอัตโนมัติ นอกจากนี้ค่าของตัวแปรจะถูกแสดงที่หน้าต่าง Environment อีกด้วย]

- เราสามารถสร้างตัวแปรจากการคำนวณตัวแปรอื่นได้ เช่น:



[เรากำหนดให้ $\text{saving} = \text{income} - \text{expense}$ แล้ว $50000 - 30000 = 20000$ ค่าของตัวแปร saving จึงเท่ากับ 20000 นั่นเอง]

-การลบตัวแปร ให้ใช้ `rm(ชื่อตัวแปร)` ในการ remove ตัวแปรออก เช่น:



[สังเกตว่าหลังจากที่เราลบตัวแปร saving ออก ก็จะไม่มีค่าในหน้าต่าง Environment อีก]

-เราสามารถลบค่าตัวแปรที่เรา Assign ไว้ทั้งหมดออกได้ด้วยการกดที่ไอคอนไม้กวาดในหน้าต่าง Environment

-ข้อดีของการเขียนโค้ดคือ มัน Reproducible (ทำซ้ำได้)

-วิธีการ Comment ใน R เราจะใช้ Hash (#) เช่น:

```
1 #Create Variables
2 income <- 50000
3 expense <- 30000
4 saving <- income - expense
5
6 #Remove Variables
7 rm(saving)
```

-ไฟล์ที่เรา Save จะไฟล์ใน Working Directory ของเรา

Lesson 3: Compare Values

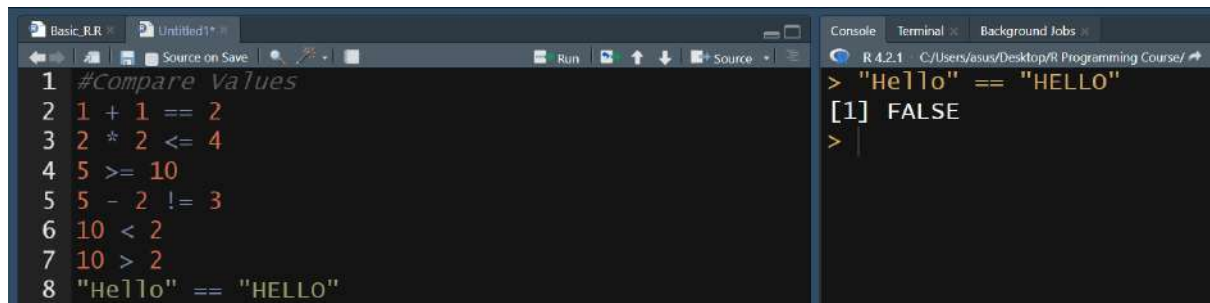
-เราสามารถเขียน Operators เหล่านี้เพื่อเปรียบเทียบทั้งสองฝั่งของสมการได้ใน R:

1. >
2. ≥
3. <
4. ≤
5. == (เท่ากับ)
6. ≠ (ไม่เท่ากับ !=)

*ถ้าใช้ = ตัวเดียวจะเป็นการประกาศตัวแปร เราสามารถใช้ R ประกาศตัวแปรได้ 2 แบบคือ ← และ =

-ผลลัพธ์ของการ Compare ค่าจะได้ TRUE (จริง) หรือ FALSE (เท็จ)

-เราสามารถ Compare Text หรือ Characters ได้เช่นกัน เช่น:



```
1 #Compare Values
2 1 + 1 == 2
3 2 * 2 <= 4
4 5 >= 10
5 5 - 2 != 3
6 10 < 2
7 10 > 2
8 "Hello" == "HELLO"
```

```
> "Hello" == "HELLO"
[1] FALSE
>
```

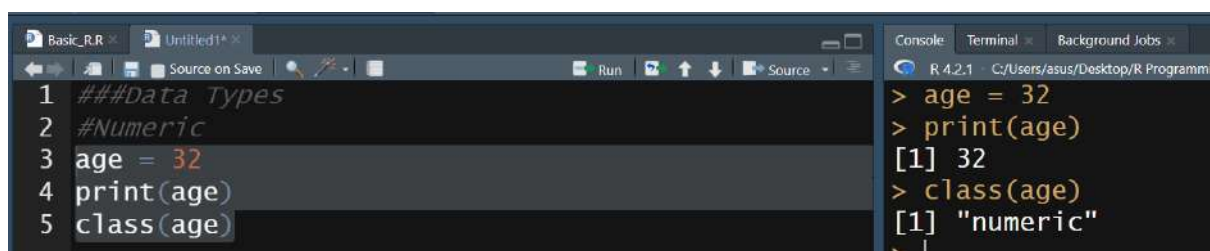
[R เป็นภาษาที่ Case Sensitive ถ้าตัวพิมพ์เล็กหรือตัวพิมพ์ใหญ่ไม่เหมือนกัน จะไม่นับว่าเหมือนกัน]

Lesson 4: Data Types

-ประเภทข้อมูลที่ Data Analyst ใช้หลัก ๆ จะมี 5 ประเภท คือ

1. Numeric (ตัวเลข)
2. Character (ตัวอักษร)
3. Logical (ค่าความจริง)
4. Factor (ตัวแปรเชิงสถิติ)
5. Date/Time (วันที่และเวลา)

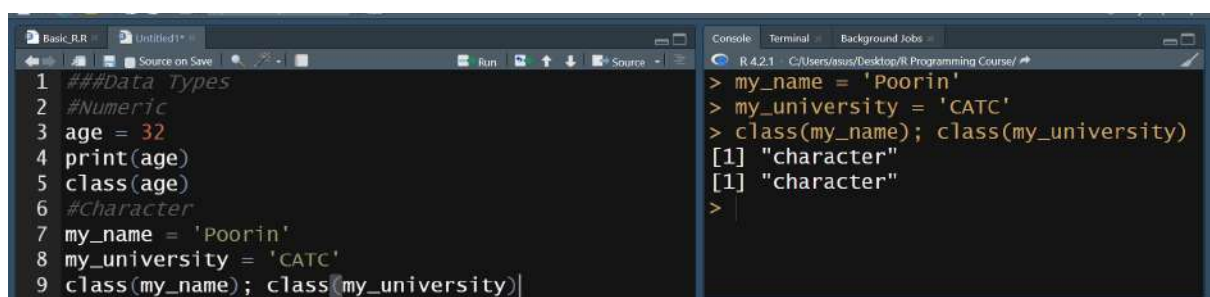
-ตัวอย่างค่าที่เป็น Numeric:



```
1 ###Data Types
2 #Numeric
3 age = 32
4 print(age)
5 class(age)
```

```
> age = 32
> print(age)
[1] 32
> class(age)
[1] "numeric"
>
```

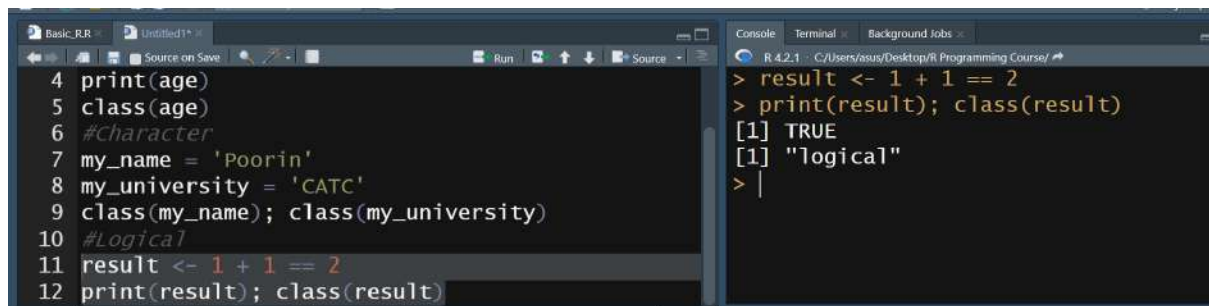
-ตัวอย่างค่าที่เป็น Character:



```
1 ###Data Types
2 #Numeric
3 age = 32
4 print(age)
5 class(age)
6 #Character
7 my_name = 'Poorin'
8 my_university = 'CATC'
9 class(my_name); class(my_university)
```

```
> my_name = 'Poorin'
> my_university = 'CATC'
> class(my_name); class(my_university)
[1] "character"
[1] "character"
>
```


-ตัวอย่างค่าที่เป็น Logical:



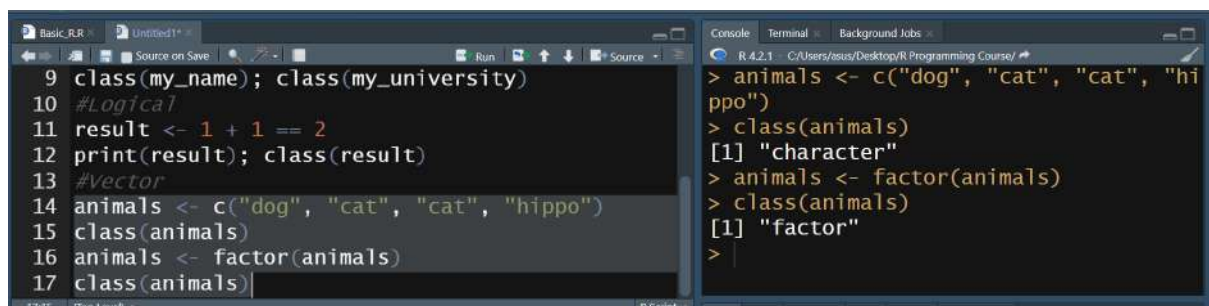
```
4 print(age)
5 class(age)
6 #Character
7 my_name = 'Poorin'
8 my_university = 'CATC'
9 class(my_name); class(my_university)
10 #Logical
11 result <- 1 + 1 == 2
12 print(result); class(result)
```

```
> result <- 1 + 1 == 2
> print(result); class(result)
[1] TRUE
[1] "logical"
> |
```

[เราสามารถเขียน T ให้ระบบรู้ว่าเป็นค่า TRUE ได้ ส่วนค่า False ก็เขียน F]

-c() คือการ Concatenate ใช้ในการสร้าง Array ของค่าต่าง ๆ

-เราสามารถใช้ factor() ในการเปลี่ยนประเภทของข้อมูลให้เป็น factor เพื่อจัดการข้อมูลที่เป็นกลุ่ม เช่น:

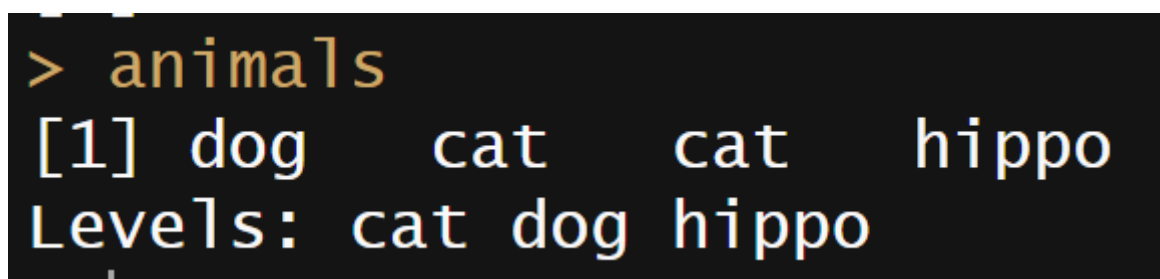


```
9 class(my_name); class(my_university)
10 #Logical
11 result <- 1 + 1 == 2
12 print(result); class(result)
13 #Vector
14 animals <- c("dog", "cat", "cat", "hippo")
15 class(animals)
16 animals <- factor(animals)
17 class(animals)
```

```
> animals <- c("dog", "cat", "cat", "hippo")
> class(animals)
[1] "character"
> animals <- factor(animals)
> class(animals)
[1] "factor"
> |
```

[จะสังเกตว่า ก่อนที่เราจะครอบตัวแปร animals ด้วย factor มันจะเป็น character และเราใช้ factor ในการเปลี่ยนประเภทของตัวแปรค่าให้กลายเป็นประเภทใหม่ จาก character → factor]

-หลังจากที่เปลี่ยนค่าของตัวแปรเป็น factor จะไม่มี “ ” ครอบอีก



```
> animals
[1] dog    cat    cat    hippo
Levels: cat dog hippo
```

-ตัวอย่างค่าที่เป็น Date/Time:

```
11 result <- 1 + 1 == 2
12 print(result); class(result)
13 #Vector
14 animals <- c("dog", "cat", "cat", "hippo")
15 class(animals)
16 animals <- factor(animals)
17 class(animals)
18 Sys.time()
```

```
> Sys.time()
[1] "2022-12-22 00:57:56 CST"
```

[สังเกตว่าเราใช้ Sys.time เพื่อเรียกวันที่และเวลาปัจจุบันในคอมพิวเตอร์ พร้อมกับแสดง Timezone ในเวลาเดียวกัน]

```
11 result <- 1 + 1 == 2
12 print(result); class(result)
13 #Vector
14 animals <- c("dog", "cat", "cat", "hippo")
15 class(animals)
16 animals <- factor(animals)
17 class(animals)
18 Time_now <- Sys.time()
19 class(Time_now)
```

```
> Sys.time()
[1] "2022-12-22 00:57:56 CST"
> Time_now <- Sys.time()
> class(Time_now)
[1] "POSIXct" "POSIXt"
```

[ถ้าเห็น POSIXct หรือ POSIXt ให้รู้ว่าเป็นตัวแปรประเภทวันที่และเวลา]

Lesson 5: Convert Data Types

-เราสามารถเปลี่ยนประเภทของข้อมูลได้ เช่น:

```
1 ##Convert Data Types
2 x <- 100
3 class(x)
4 char_x <- as.character(x)
5 char_x; class(char_x)
6 num_x <- as.numeric(char_x)
7 num_x; class(num_x)
```

```
> x <- 100
> class(x)
[1] "numeric"
> char_x <- as.character(x)
> char_x; class(char_x)
[1] "100"
[1] "character"
> num_x <- as.numeric(char_x)
> num_x; class(num_x)
[1] 100
[1] "numeric"
```

[เราใช้ as.ประเภทของตัวแปรที่เราต้องการจะเปลี่ยน เพื่อเปลี่ยนประเภทของตัวแปรนั้น ๆ ได้ และเราสามารถเปลี่ยนประเภทของตัวแปรไปมาได้ตามต้องการ]

*True มีค่า = 1 ส่วน False มีค่า = 0


```
4 char_x <- as.character(x)
5 char_x; class(char_x)
6 num_x <- as.numeric(char_x)
7 num_x; class(num_x)
8
9 as.logical(0)
10 as.logical(1)
11 as.numeric(TRUE)
12 as.numeric(FALSE)
```

```
> as.logical(0)
[1] FALSE
> as.logical(1)
[1] TRUE
> as.numeric(TRUE)
[1] 1
> as.numeric(FALSE)
[1] 0
>
```

[เราสามารถสลับไปมาระหว่าง Numeric และ Logical ได้]

Lesson 6: Vector

-Data Structures ในงาน Data Science จะมี 4 ประเภทหลัก ๆ คือ

1. Vector
2. Matrix
3. List
4. DataFrame

-ตัวอย่างข้อมูลที่เป็น Vector:

```
1 #Vector
2 1:5
3 10:15
4 |
```

```
> #Vector
> 1:5
[1] 1 2 3 4 5
> 10:15
[1] 10 11 12 13 14 15
>
```

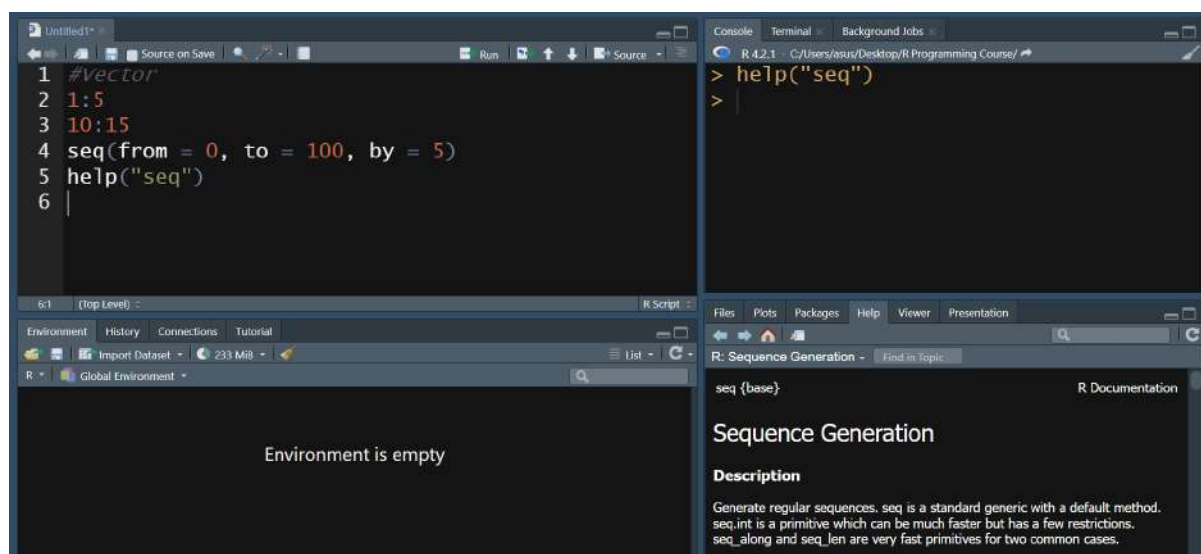
-เราสามารถใช้ seq() ทำ sequence generation ได้ เช่น:

```
1 #Vector
2 1:5
3 10:15
4 seq(from = 1, to = 100, by = 2)|
```

```
> seq(from = 1, to = 100, by = 2)
[1] 1 3 5 7 9 11 13 15 17 19
[11] 21 23 25 27 29 31 33 35 37 39
[21] 41 43 45 47 49 51 53 55 57 59
[31] 61 63 65 67 69 71 73 75 77 79
[41] 81 83 85 87 89 91 93 95 97 99
```

[เรา Start ค่าที่เลข 1 และ Step ขึ้นทีละ 2 โดยมี Stop ที่ 100]

*ถ้าเจอ Function ที่เราไม่คุ้นเคย เราสามารถใช้ help("ชื่อ function") เพื่อขอคำอธิบายเพิ่มเติมเกี่ยวกับ function นั้น ๆ ได้ เช่น:



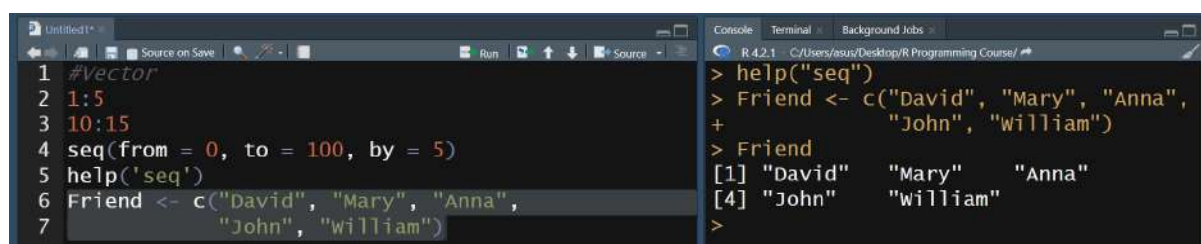
The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```
1 #Vector
2 1:5
3 10:15
4 seq(from = 0, to = 100, by = 5)
5 help("seq")
6
```

The console on the right shows the command `> help("seq")` and the resulting help page for the `seq` function. The help page includes the title "Sequence Generation" and a description: "Generate regular sequences. seq is a standard generic with a default method. seq.int is a primitive which can be much faster but has a few restrictions. seq_along and seq_len are very fast primitives for two common cases."

[สังเกตที่มุมขวาล่าง จะอธิบาย function อย่างละเอียด]

-function ที่ใช้สร้าง Vector บ่อยที่สุดใน R คือ `c()` เช่น:



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

```
1 #Vector
2 1:5
3 10:15
4 seq(from = 0, to = 100, by = 5)
5 help('seq')
6 Friend <- c("David", "Mary", "Anna",
7             "John", "william")
```

The console on the right shows the command `> help("seq")` and the resulting help page for the `seq` function. Below the help page, the command `> Friend <- c("David", "Mary", "Anna", "John", "william")` is shown, followed by the output of `> Friend`:

```
[1] "David" "Mary"  "Anna"
[4] "John"  "william"
```

[Vector จะเก็บข้อมูลได้แต่ข้อมูลประเภทเดียวกันทั้งหมดเท่านั้น (1 ประเภท) คณะประเภทข้อมูลไม่ได้]

Lesson 7: Matrix

-Matrix เป็นข้อมูลที่มี 2 Dimensions เราสามารถใช้ `dim()` แล้วพิมพ์ชื่อตัวแปรในการสร้าง Matrix ได้ เช่น:

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

```
> 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
```

-อีกวิธีหนึ่งคือการใช้ `matrix()` สร้าง Matrix เช่น:

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

```
> matrix(1:25, ncol = 5)
      [,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
```

[กำหนดให้ช่วงตัวเลข = 1 ถึง 25 และจำนวน column = 5]

-เราสามารถกำหนดจำนวน row ได้เช่นกัน นอกเหนือจากการกำหนดจำนวน column เช่น:

```
1 #Matrix
2 x <- 1:25
3 length(x)
4 dim(x) <- c(5,5)
5
6 matrix(1:25, ncol = 5)
7
8 matrix(1:6, ncol = 3, nrow = 2, byrow = T)
9
```

```
> matrix(1:6, ncol = 3, nrow = 2, byrow = T)
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

-และ เราสามารถกำหนดค่าตัวแปรเป็น Matrix ได้ ด้วยการ Assign Matrix ให้ตัวแปรด้วย ←

-ถ้าเราเอาตัวเลขไปบวกกับตัวแปรที่เป็น Matrix ระบบจะบวกค่าทั้งหมดใน Matrix ใน R จะเรียกว่า Element Wise Computation (สมมติว่า $m1 * 4$ ค่าทั้งหมดใน Matrix `m1` ก็จะถูกคูณด้วย 4)

Lesson 8: List

-List เป็นประเภทข้อมูลใน R ที่สามารถเก็บค่าได้หลายประเภท

-เราสามารถสร้าง List ได้ด้วย `list()` เช่น:

```

2 my_name = 'Pooh'
3 my_friend = c('Tanat', 'Sato', 'Boom')
4 m1 = matrix(1:25, ncol = 5)
5 R_is_cool = TRUE
6
7 my_list = list(item_1 = my_name,
8               item_2 = my_friend,
9               item_3 = m1,
10              item_4 = R_is_cool)

```

```

> my_list
$item_1
[1] "Pooh"

$item_2
[1] "Tanat" "Sato"  "Boom"

$item_3
     [,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

```

[สังเกตว่า List สามารถเก็บค่าได้หลายประเภท ตั้งแต่ Character, Vector Matrix, ไปจนถึง Logical]

-เราสามารถดึงเฉพาะ Item ใน List ที่เราต้องการได้ เช่น:

```

5 R_is_cool = TRUE
6
7 my_list = list(item_1 = my_name,
8               item_2 = my_friend,
9               item_3 = m1,
10              item_4 = R_is_cool)
11
12 my_list$item_3
13

```

```

> my_list$item_3
     [,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

```

[สมมติเราต้องการเฉพาะ item_3 ที่เป็น Matrix เราสามารถพิมพ์ \$ ตามด้วย item_3 ต่อท้าย my_list ได้]

Lesson 9: Data Frame

*ความรู้เรื่อง Data Frame สำคัญมาก เพราะงานของ Data Analyst โดยส่วนมากจะทำงานกับ Structured Data (ข้อมูลที่ถูกจัดเก็บอย่างเป็นระเบียบ) เช่น ข้อมูลใน Google Sheets หรือ ข้อมูลที่เราดึงมาจาก Database

-เราจะใช้ data.frame() ในการสร้าง Data Frame เช่น:

```

1 #DataFrame
2 game = c('Genshin', 'Granblue', 'Nikke')
3 year_played = c(2, 6, 1)
4 region = c('China', 'Japan', 'South Korea')
5 good_gacha = c(T, F, F)
6
7 df = data.frame(game,
8               year_played,
9               region,
10              good_gacha)

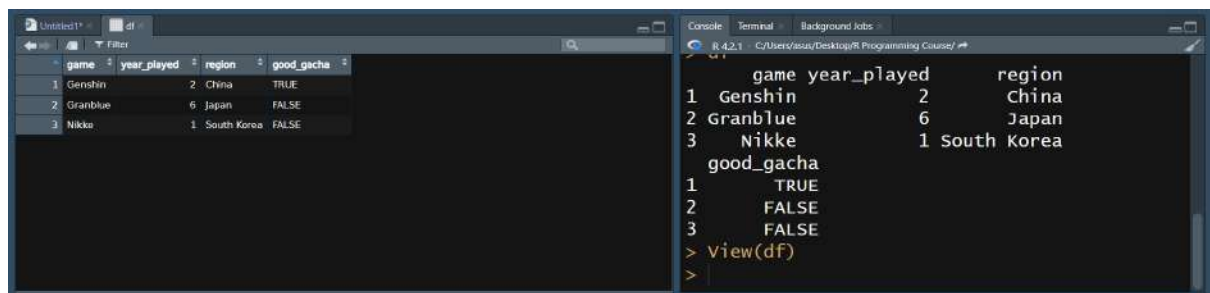
```

```

> df
  game year_played region
1 Genshin         2  China
2 Granblue         6  Japan
3 Nikke           1 South Korea
  good_gacha
1      TRUE
2     FALSE
3     FALSE

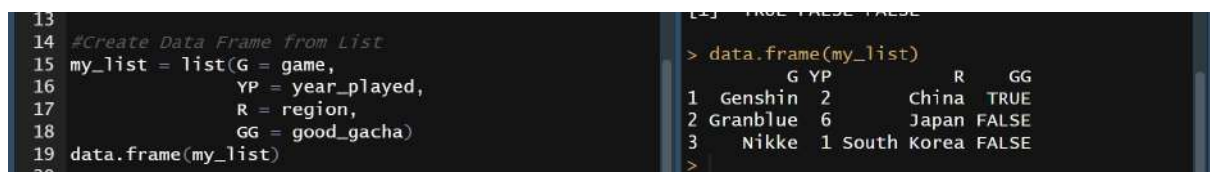
```

-มีอีก function หนึ่งในการมองข้อมูลให้ดูเป็น Data Frame มากขึ้น คือ View() เช่น:



[สังเกตว่าข้อมูลจะดูเรียบร้อยกว่าดูผ่าน Console]

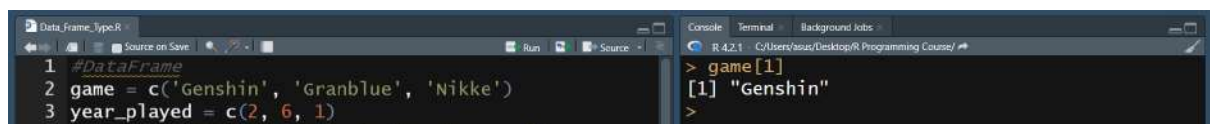
-เราสามารถสร้าง Data Frame จาก List ได้เช่นกัน โดยที่เราสามารถตั้งชื่อให้แต่ละ Item ใน List ได้ เป็นอีกวิธีหนึ่งในการสร้าง Data Frame เช่น:



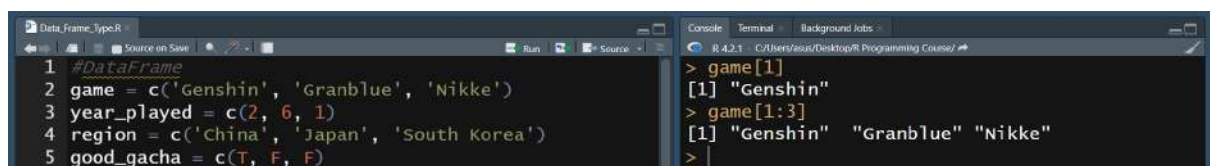
Lesson 10: Subset

-Subset คือการดึง Data ออกจาก Data Structure ทำได้หลายวิธี

-Subset by Position:



-เราสามารถทำเป็น Slicing ได้เช่นกัน เช่น:



-ถ้าอยากเลือกค่าแบบกระโดด ให้ใช้ c() ช่วย เช่น game[c(1, 3, 5)]

-Subset by Condition:


```
Data_Frame_Type.R
1 #DataFrame
2 game = c('Genshin', 'Granblue', 'Nikke')
3 year_played = c(2, 6, 1)
4 region = c('China', 'Japan', 'South Korea')
5 good_gacha = c(T, F, F)
6

Console
R 4.2.1 - C:/Users/asus/Desktop/R Programming Course/
> game[1]
[1] "Genshin"
> game[1:3]
[1] "Genshin" "Granblue" "Nikke"
> year_played[ year_played > 1]
[1] 2 6
```

[เราสามารถกรองเฉพาะข้อมูลที่ต้องการได้ด้วยการตั้งเงื่อนไข]

-Subset by Name (ใช้ name() wrap array ที่ต้องการ)เช่น:

```
Data_Frame_Type.R
1 #DataFrame
2 game = c('Genshin', 'Granblue', 'Nikke')
3 year_played = c(2, 6, 1)
4 region = c('China', 'Japan', 'South Korea')
5 good_gacha = c(T, F, F)
6

Console
R 4.2.1 - C:/Users/asus/Desktop/R Programming Course/
> names(game) = region
> game
      China      Japan South Korea
"Genshin" "Granblue"    "Nikke"
```

-สมมติว่า เราอยากจะรู้ว่าญี่ปุ่น (Japan) เป็น region ของเกมไหน เราสามารถพิมพ์ game['Japan'] เพื่อหาเกมที่ผลิตในญี่ปุ่นได้ เช่น:

```
R 4.2.1 - C:/Users/asus/Desktop/R Programming Course/
> names(game) = region
> game
      China      Japan South Korea
"Genshin" "Granblue"    "Nikke"
> game['Japan']
      Japan
"Granblue"
>
```

-สำหรับการทำ Subset ของ Data Frame เราจะต้องระบุ row และ column ที่เราต้องการ (เพราะ Data Frame มี 2 มิติ) เช่น สมมติเราอยากได้ข้อมูลของ row ที่ 1 และ column ที่ 2 เราสามารถทำได้ดังนี้ (Subset by Position):

```
Data_Frame_Type.R
1 #DataFrame
2 game = c('Genshin', 'Granblue', 'Nikke')
3 year_played = c(2, 6, 1)
4 region = c('China', 'Japan', 'South Korea')
5 good_gacha = c(T, F, F)
6

Console
R 4.2.1 - C:/Users/asus/Desktop/R Programming Course/
> df[1, 2]
[1] 2
>
```

| game | year_played | region | good_gacha |
|------------|-------------|-------------|------------|
| 1 Genshin | 2 | China | TRUE |
| 2 Granblue | 6 | Japan | FALSE |
| 3 Nikke | 1 | South Korea | FALSE |

```

> View(df)
> df[1:1, 3:4]
  region good_gacha
1  China      TRUE
> df[1:1, 2:4]
  year_played region good_gacha
1           2  China      TRUE
> df[1:1, 1:4]
  game year_played region good_gacha
1 Genshin         2  China      TRUE

```

-ตัวอย่างการ Subset by Name ใน Data Frame:

| game | region |
|------------|-------------|
| 1 Genshin | China |
| 2 Granblue | Japan |
| 3 Nikke | South Korea |

```

> df[, c('game', 'region')]
  game region
1 Genshin  China
2 Granblue  Japan
3 Nikke    South Korea

```

[ดึงข้อมูลทั้งหมดจาก column game และ region]

-ตัวอย่างการ Subset by Condition ใน Data Frame:

| game | year_played | region | good_gacha |
|------------|-------------|-------------|------------|
| 2 Granblue | 6 | Japan | FALSE |
| 3 Nikke | 1 | South Korea | FALSE |

```

> df[df$good_gacha == FALSE, ]
  game year_played region good_gacha
2 Granblue         6  Japan      FALSE
3 Nikke          1 South Korea FALSE

```

[ดึงข้อมูลจาก row ที่ good_gacha == FALSE]