

R 103

R สามารถอ่านไฟล์ข้อมูลได้หลายประเภท เช่น

`.txt .csv .xlsx .json .db` หรือ `google sheet` และ SQL databases



ติดตั้ง library ได้ด้วยคำสั่ง `install.packages()` สามารถรันใน RStudio Console ได้เลย

```
# install packages
install.packages(c("readr",
                  "readxl",
                  "googlesheet4",
                  "jsonlite",
                  "dplyr",
                  "sqldf",
                  "RSQLite"))

# load library
library(readr)
library(readxl)
library(googlesheet4)
library(jsonlite)
library(dplyr)
library(sqldf)
library(RSQLite)
```

result

```
1 # install packages
2 install.packages(c("readr",
3                   "readxl",
4                   "googlesheets4",
5                   "jsonlite",
6                   "dplyr",
7                   "sqldf",
8                   "RSQLite"))
9
10 # load library
11 library(readr)
12 library(readxl)
13 library(googlesheets4)
14 library(jsonlite)
15 library(dplyr)
16 library(sqldf)
17 library(RSQLite)
18
```

```
> library(sqldf)
Loading required package: gsubfn
Loading required package: proto
Loading required package: RSQLite
Warning message:
In fun(libname, pkgname) : couldn't connect to display ":0"
> library(RSQLite)
>
```

Name	Description	Version
skimr	Compact and Flexible Summaries of Data	2.1.5

สามารถกด install ของ packages ได้เลย

[Text File](#)

[CSV](#)

[Excel File](#)

[Google Sheets](#)

[JSON](#)

[Bind Rows \(UNION ALL\)](#)

[Bind Cols\(≠JOIN\)](#)

[SQL](#)

[SQLite](#)

[How to save data in R](#)

[How to install R packages](#)

[Lesson 1: Text File](#)

[Lesson 2: CSV](#)

[Lesson3 : Excel File](#)

[Lesson 4: Google Sheets](#)

[Lesson 5: JSON](#)

[Lesson 6: Bind Rows \(UNION ALL\)](#)

[Lesson 7: Bind Cols \(≠ JOIN\)](#)

[Lesson 8: SQL](#)

[Lesson 9: SQLite](#)

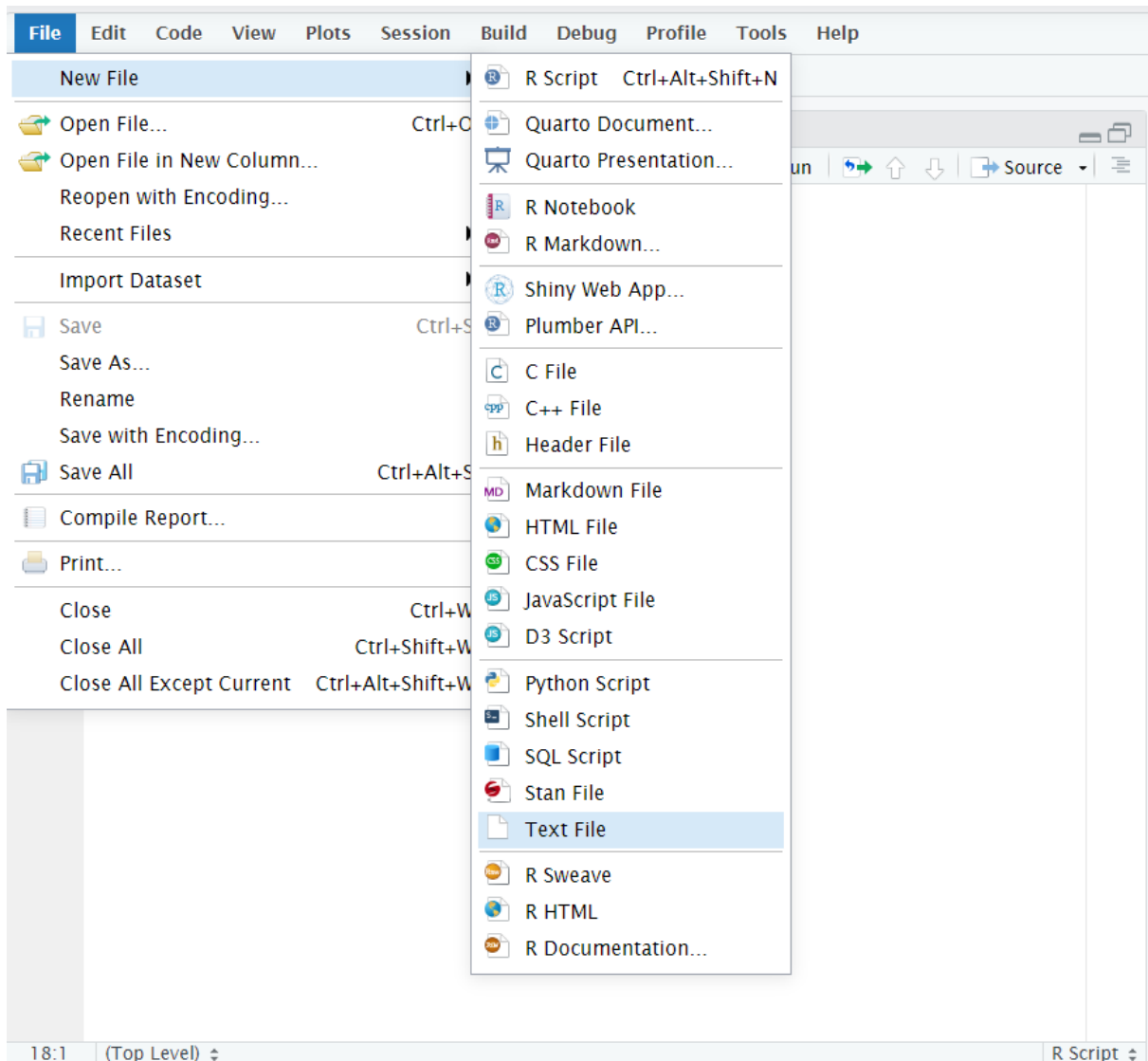
[Lesson 10: How to save data in R](#)

Text File



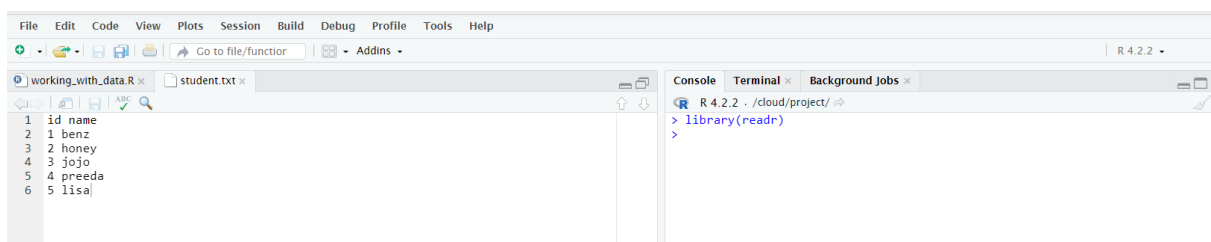
`read_table()` อ่านไฟล์ .txt ที่ใช้ whitespace เป็นตัวคั่นระหว่างcolumn(delimiter)

result



create text file

result



รับคำสั่งที่console ในที่นี้คือ `library(readr)` ก่อนใช้งานเสมอ

The screenshot shows RStudio with a text file 'student.txt' loaded into a data frame named 'student'. The data is as follows:

id	name
1	benz
2	honey
3	jojo
4	preeda
5	lisa

The R code in the console is:

```
R 4.2.2 - /cloud/project/
> library(readr)
> read_table("student.txt")

# A tibble: 5 × 2
  id name
<dbl> <chr>
1     1 benz
2     2 honey
3     3 jojo
4     4 preeda
5     5 lisa

> student <- read_table("student.txt")

# Column specification
cols(
  id = col_double(),
  name = col_character()
)

> view(student)
Error in view(student) : could not find function "view"
> View(student)
> View(student)
> |
```

`read_table("ชื่อtext file.txt")` คือการอ่านไฟล์textนั้นขึ้นมา, `View()` ใช้เพื่อแสดงข้อมูลเป็นtable

CSV



`read_csv()` ใช้อ่านไฟล์ csv(ย่อมาจาก comma separated values) เป็น common data format ที่ data analyst ใช้ประจำ

result

The screenshot shows RStudio with a CSV file 'student.csv' loaded into a data frame named 'student2'. The data is as follows:

id	name
1	benz
2	honey
3	jojo
4	preeda
5	lisa

The R code in the console is:

```
R 4.2.2 - /cloud/project/
> read_csv("student.csv")
Rows: 5 Columns: 2
# Column specification
Delimiter: ","
chr (1): name
dbl (1): id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this mess-
age.
# A tibble: 5 × 2
  id name
<dbl> <chr>
1     1 benz
2     2 honey
3     3 jojo
4     4 preeda
5     5 lisa

> student2 <- read_csv("student.csv")
Rows: 5 Columns: 2
# Column specification
Delimiter: ","
chr (1): name
dbl (1): id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this mess-
age.
> View(student2)
> |
```

ดูตารางtable ไม่ต้องพิมพ์ View() สามารถกดที่ student2 ใน Environment ได้เลย

Excel File



`read_excel()` ใช้อ่านไฟล์ Excel.xlsx อ่านทีละ sheet ต้องโหลด
`library(readxl)` ก่อนทุกครั้งที่จะใช้งาน

download

students.xlsx

result

The screenshot shows the RStudio interface. The Environment pane on the left displays a data frame named 'econ_student' with 5 observations and 3 variables. The Console pane on the right shows the R code used to read the Excel file and the resulting data frames.

student_id	name	major
1	toy	economics
2	john	economics
3	mary	economics
4	anna	economics
5	lisa	economics

```

> read_excel("students.xlsx", sheet="Data")
# A tibble: 5 × 3
  student_id name      major
  <dbl> <chr> <chr>
1      11 andrew  data
2      12 yan     data
3      13 yoshua data
4      14 jane   data
5      15 nanny  data
> read_excel("students.xlsx", sheet="Economics")
# A tibble: 5 × 3
  student_id name      major
  <dbl> <chr> <chr>
1         1 toy     economics
2         2 john    economics
3         3 mary    economics
4         4 anna    economics
5         5 lisa     economics
> econ_student <- read_excel("students.xlsx", sheet="Economics")
> View(econ_student)
  
```

รู้ชื่อsheetใส่ชื่อsheet รู้ตำแหน่งใส่ตำแหน่ง

loop เปิดข้อมูลทุกหน้าของข้อมูลใน sheet

```

result <- list()

for (i in 1:3) {
  result[[i]] <- read_excel("students.xlsx", sheet=i)
}

result[[1]]
result[[2]]
result[[3]]
  
```

result

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

```
1 result <- list()
2
3 for (i in 1:3) {
4   result[[i]] <- read_excel("students.xlsx", sheet=i)
5 }
6
```

The console on the right shows the output of the script. It displays three tibbles, one for each sheet (1, 2, and 3). The first two tibbles are 5x3, and the third is 5x3. The data is as follows:

Sheet	student_id	name	major
1	2	john	economics
1	3	mary	economics
1	4	anna	economics
1	5	lisa	economics
2	6	henry	business
2	7	david	business
2	8	jisoo	business
2	9	jenny	business
2	10	kevin	business
3	11	andrew	data
3	12	yan	data
3	13	yoshua	data
3	14	jane	data
3	15	nanny	data

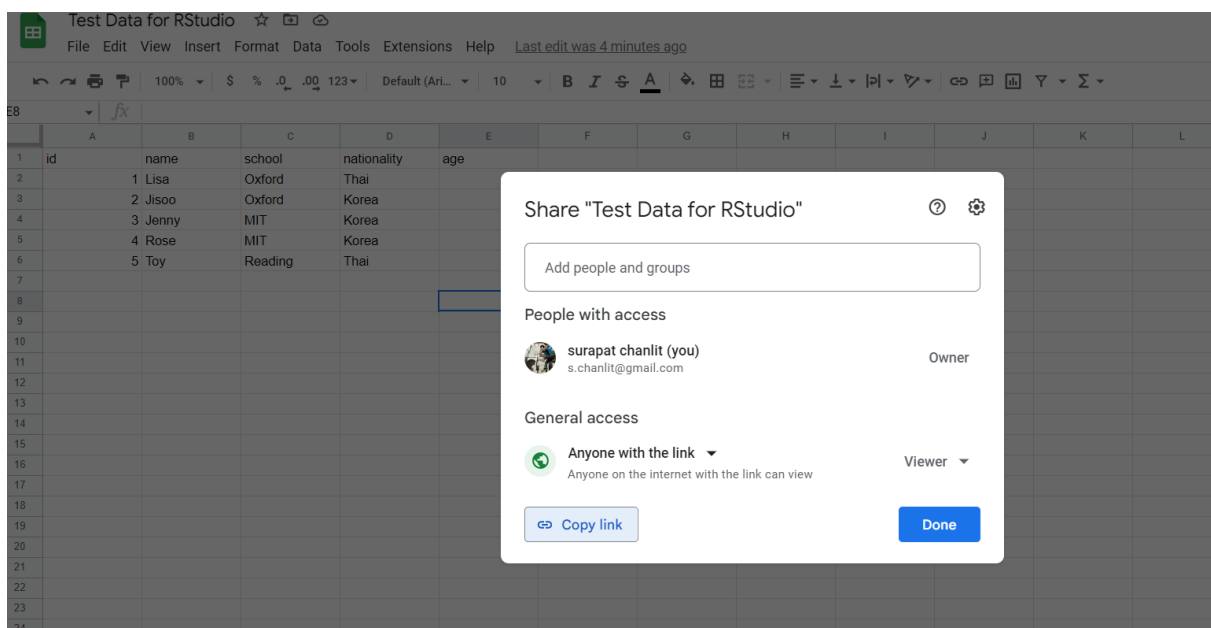
i ย่อมาจาก index เมื่อ print result ใน console `[[i]]` i แทนหน้าของ sheet นั้นๆ

Google Sheets



`read_sheet()` ใช้อ่าน data จาก Google Sheet ถ้า link เป็น public ให้ใช้คำสั่ง `gs4_deauth()` ก่อน (ลิ่งที่จะอ่านไฟล์เป็น public link ไม่มีการ login)

Ex เริ่มจากสร้างdata base ในgoogle sheet



กด share มุมขวา check access

result

The screenshot shows the RStudio interface. The top-left pane displays a data frame 'df' with 5 rows and 5 columns: id, name, school, nationality, and age. The data is as follows:

id	name	school	nationality	age
1	Lisa	Oxford	Thai	25
2	Jisoo	Oxford	Korea	22
3	Jenny	MIT	Korea	24
4	Rose	MIT	Korea	28
5	Toy	Reading	Thai	29

The bottom-left pane shows the Environment tab with 'df' listed as a data frame with 5 observations and 5 variables. The console on the right shows the execution of the following R code:

```
> library(googleSheets4)
Warning message:
In fun(libname, pkgname) : couldn't connect to display ":0"
> url <- "https://docs.google.com/spreadsheets/d/1JHZnomxDceb_GCy994y6Fr-MURMm7w_zPEEcVrtsBA8"
> gs4_deauth()
>
Connected to your session in progress, last started 2023-Feb-13 06:39:17 UTC
(6 minutes ago)
> read_sheet(url, sheet="student")
✓ Reading from Test Data for RStudio.
✓ Range 'student'.
# A tibble: 5 × 5
  id name school nationality age
<dbl> <chr> <chr> <chr> <dbl>
1     1 Lisa Oxford Thai      25
2     2 Jisoo Oxford Korea    22
3     3 Jenny MIT Korea    24
4     4 Rose MIT Korea    28
5     5 Toy Reading Thai     29
>
> df <- read_sheet(url, sheet="student")
✓ Reading from Test Data for RStudio.
✓ Range 'student'.
> View(df)
```

ใช้ `gs4_deauth()` เนื่องจาก sheet เป็น public

JSON

JSON = JavaScript Object Notation



`fromJSON()` ใช้อ่านไฟล์ .json เข้ามาเป็น list ใน R แล้วค่อยฟังก์ชัน `data.frame()` เพื่อเปลี่ยน list -> data frame ได้

key ใน JSON ต้องเขียนใน double quote เท่านั้น

```
library(jsonlite)
```

```
bp <- data.frame(fromJSON("blackpink.json"))
```

```
View(bp)
```

result

```

1 {
2   "id": [1,2,3,4,5],
3   "name": ["toy", "john", "Mary", "lisa", "jisoo"],
4   "love_singing": [true, false, false, true, true]
5 }

```

```

R 4.2.2 . /cloud/project/
> library(jsonlite)
> 
> fromJSON("blackpink.json")
$id
[1] 1 2 3 4 5

$name
[1] "toy" "john" "Mary" "lisa" "jisoo"

$love_singing
[1] TRUE FALSE FALSE TRUE TRUE

> 
> data <- fromJSON("blackpink.json")
> data
$id
[1] 1 2 3 4 5

$name
[1] "toy" "john" "Mary" "lisa" "jisoo"

$love_singing
[1] TRUE FALSE FALSE TRUE TRUE

> data.frame(data)
  id name love_singing
1  1  toy          TRUE
2  2  john         FALSE
3  3  Mary          FALSE
4  4  lisa          TRUE
5  5  jisoo         TRUE
> bp <- data.frame(fromJSON("blackpink.json"))
> 
> bp
  id name love_singing
1  1  toy          TRUE

```

data.frame คือการเปลี่ยน data ให้เป็น column rows

Bind Rows (UNION ALL)



`bind_rows()` เหมือนกับ `UNION ALL` SQL

ใช้ไฟล์ *students.xlsx*

result

```

1 ## read excel file
2 econ <- read_excel("students.xlsx", sheet=1)
3 business <- read_excel("students.xlsx", sheet=2)
4 data <- read_excel("students.xlsx", sheet=3)
5 
6 ## bind_rows == SQL UNION ALL
7 list_df <- list(econ, business, data)
8 full_df <- bind_rows(list_df)

```

```

R 4.2.2 . /cloud/project/
3      mary economics
4      anna economics
5      lisa economics
6      henry business
7      david business
8      jisoo business
9      jenny business
10     kevin business
11     andrew data
12     yan data
13     yoshua data
14     jane data
15     nanny data

> list_df <- list(econ, business, data)
> full_df <- bind_rows(list_df)
> full_df
# A tibble: 15 x 3
  student_id name major
  <dbl> <chr> <chr>
1         1 toy economics
2         2 john economics
3         3 mary economics
4         4 anna economics
5         5 lisa economics
6         6 henry business
7         7 david business
8         8 jisoo business
9         9 jenny business
10        10 kevin business
11        11 andrew data
12        12 yan data
13        13 yoshua data
14        14 jane data
15        15 nanny data

```

bind_rows คือการผูกdataframeเข้าด้วยกัน ถ้ามีเยอะๆควรเขียนสูตรlistขึ้นมาก่อน

Bind Cols(≠JOIN)



`bind_col()` ไม่เท่ากับ `join` เพราะว่าไม่จำเป็นต้องใช้keyเอาแค่data frameมาต่อกัน
ซ้ายขวา

result

```
## bind_cols() != JOIN
df1 <- data.frame(
  id = 1:5,
  name = c("John", "Marry", "Anna",
            "David", "Lisa")
)
df2 <- data.frame(
  city = c( rep("BKK",3), rep("LONDON",2) ),
  country = c( rep("TH", 3), rep("UK",2) )
)
df2 <- data.frame(
  id = 1:5,
  city = c( rep("BKK",3), rep("LONDON",2) ),
  country = c( rep("TH", 3), rep("UK",2) )
)

> df2
  city country
1 BKK      TH
2 BKK      TH
3 BKK      TH
4 LONDON    UK
5 LONDON    UK
> bind_cols(df1, df2)
  id name city country
1 1 John  BKK      TH
2 2 Marry BKK      TH
3 3 Anna  BKK      TH
4 4 David LONDON   UK
5 5 Lisa LONDON   UK
> df2 <- data.frame(
+   id = 1:5,
+   city = c( rep("BKK",3), rep("LONDON",2) ),
+   country = c( rep("TH", 3), rep("UK",2) )
+ )
> df2
  id city country
1 1 BKK      TH
2 2 BKK      TH
3 3 BKK      TH
4 4 LONDON    UK
5 5 LONDON    UK
> left_join(df1, df2, by="id")
  id name city country
1 1 John  BKK      TH
2 2 Marry BKK      TH
3 3 Anna  BKK      TH
4 4 David LONDON   UK
5 5 Lisa LONDON   UK
```

join ใช้ key (id = 1:5) ส่วน bind ไม่ใช้

WHEN to JOIN



ในความเป็นจริง `bind_col()` ไม่ได้ใช้บ่อยเท่า `join` เขียน `join` ดีกว่าตรงที่ match id
ของสอง dataframes หรือมากกว่าได้ เช่น `customer_id` เป็นต้น

ใน R เขียน `join` ได้หลายแบบกว่า SQL

`inner_join`, `lef_join`, `right_join`, `full_join`, `anti_join`, `semi_join`

SQL



`sqldf()` ใช้เพื่อจัดการ dataframe ที่อยู่ใน R

download

[school.zip](#)

result

```
1 # load library sqldf
2 library(sqldf)
3 library(readr)
4
5 school <- read_csv("school.csv")
6
7 sqldf("select * from school;")
8
9 sqldf("select
10   avg(student),
11   sum(student)
12   from school;")
13
14 sqldf("
15   select
16     school_id,
17     school_name,
18     country
19   from school;
20 ")
21
22 sql_query <- "select * from school
23   where country = 'USA' "
24
```

```
> school
# A tibble: 5 × 5
  school_id school_name      country student top10
  <dbl>    <chr>          <chr>    <dbl> <lgL>
1         1 University of London UK      50000 FALSE
2         2 Cambridge          UK      65000 TRUE
3         3 Oxford            UK      32000 TRUE
4         4 Harvard            USA     48000 TRUE
5         5 MIT                USA     62000 TRUE

> sqldf("select * from school;")
  school_id school_name      country student top10
1         1 University of London UK      50000 FALSE
2         2 Cambridge          UK      65000 TRUE
3         3 Oxford            UK      32000 TRUE
4         4 Harvard            USA     48000 TRUE
5         5 MIT                USA     62000 TRUE

> sqldf("select
+   avg(student),
+   sum(student)
+   from school;")
  avg(student) sum(student)
1         51400      257000

> sqldf("
+   select
+     school_id,
+     school_name,
+     country
+   from school;
+ ")
  school_id school_name      country
1         1 University of London UK
2         2 Cambridge          UK
```

SQLite

ใน R ใช้ library **RSQLite** เพื่อจัดการข้อมูลใน sqlite.db file แบ่งเป็นสามขั้นตอนคือ

1. connect ot database i.e.open connection
2. get data(with SQL)
3. disconnect from database i.e.close connection



เพิ่มเติมได้ที่ <https://dbi.r-dbi.org/>

download

[chinook.db](#)

```
# load library
library(RSQLite)

# connect to SQLite database (.db file)
# 1. open connection
conn <- dbConnect(SQLite(), "chinook.db")
```

result

```
# load library
library(RSQLite)

# connect to SQLite database (.db file)
# 1. open connection
conn <- dbConnect(SQLite(), "chinook.db")

# 2. get data
dbListTables(conn)
dbListFields(conn, "customers")

df <- dbGetQuery(conn, "select * from customers where country = 'USA'")
df2 <- dbGetQuery(conn, "select * from customers where country = 'United Kingdom'")

# 3. close connection
dbDisconnect(conn)
```

```
R 4.2.2 - /cloud/project/
1      4
2      5
3      3
4      3
5      4
6      5
7      4
8      4
9      3
10     5
11     4
12     4
13     5

> df2 <- dbGetQuery(conn, "select * from customers where country = 'United Kingdom'")
> df2
  CustomerId FirstName LastName Company      Address      City
1         52      Emma      Jones <NA> 202 Hoxton Street  London
2         53      Phil      Hughes <NA>      113 Lupus St  London
3         54      Steve     Murray <NA>    110 Raeburn Pl  Edinburgh
      State      Country PostalCode      Phone      Fax
1 <NA> United Kingdom      N1 5LH +44 020 7707 0707 <NA>
2 <NA> United Kingdom      SW1V 3EN +44 020 7976 5722 <NA>
3 <NA> United Kingdom      EH4 1HH +44 0131 315 3300 <NA>
      Email      SupportRepId
1 emma.jones@hotmail.com      3
2 phil.hughes@gmail.com      3
3 steve.murray@yahoo.uk      5

> # 3. close connection
> dbDisconnect(conn)
```



เราสามารถ save data ใน R ได้สองแบบและสามารถโหลดกลับมาใช้ใหม่ได้

- `save.image()` ใช้ save objects ทั้งหมดที่อยู่ใน environment เข้าไปที่ไฟล์ `.RData`
- `saveRDS()` ใช้ save single object แยกไฟล์เดี่ยวที่ไฟล์ `.rds`

ถ้าต้องการโหลด `.RData` ใช้ฟังก์ชัน `load()`

ถ้าต้องการโหลด `.rds` ใช้ฟังก์ชัน `readRDS()`

R superpower!

-R เป็นภาษาที่ Fast Data Crunching ที่ถูกพัฒนาขึ้นมาสำหรับงาน Data โดยเฉพาะ เปิดโปรแกรมแล้วทำงานได้ทันที

-R เป็น 1 ใน 3 ภาษาโปรแกรมที่ Data Analyst ต้องรู้จักและใช้ให้เป็น

-เราสามารถใช้ R อ่านไฟล์ข้อมูลได้หลายประเภท

-เราสามารถติดตั้ง library เพิ่มเติมนอกเหนือจาก base R เพื่อช่วยในการทำงานได้ เช่น:

```
#Install Packages
install.packages(c("readr",
                  "readxl",
                  "googlesheets4",
                  "jsonlite",
                  "dplyr",
                  "sqldf",
                  "RSQLite"))
```

```
#Load Library
library(readr)
library(readxl)
library(googlesheets4)
library(jsonlite)
library(dplyr)
library(sqldf)
library(RSQLite)
```

How to install R packages

-Package หรือ Library มีความหมายเหมือนกัน

-เราสามารถติดตั้ง package ใน R ได้ 2 วิธี คือ:

1. `install.packages()`
2. Packages Tab ใน RStudio → กด install

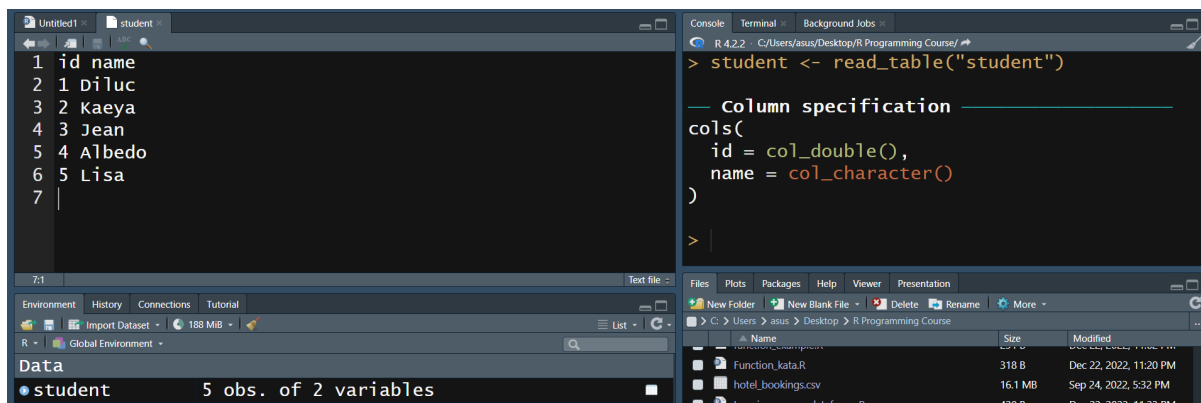
-Package ที่เราสามารถโหลดมาใช้ได้: https://cran.r-project.org/web/packages/available_packages_by_name.html

-เราสามารถกด Update Tab ใน RStudio เพื่อ Update library ได้

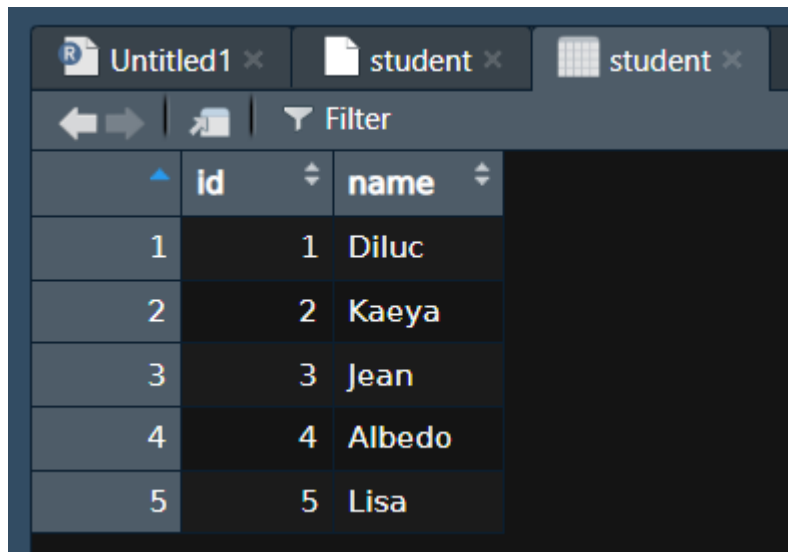
Lesson 1: Text File

-RStudio สามารถเขียน Script ได้มากกว่าภาษา R เช่น C++, Python, SQL เป็นต้น

-`read_table()` ใช้อ่าน file .txt ที่ใช้ whitespace (ช่องว่าง) เป็นตัวคั่นระหว่าง column (delimiter) [ก่อนใช้ ให้โหลด library readr มาก่อน] เช่น:



[หลังจากที่เรา assign ตัว text file ของเราเป็นตัวแปรแล้ว เราสามารถกดที่ไอคอนตารางตรง Data เพื่อให้แสดงผลออกมาเป็น Table ได้ หรือจะใช้ `View(student)` ก็ได้เช่นกัน]

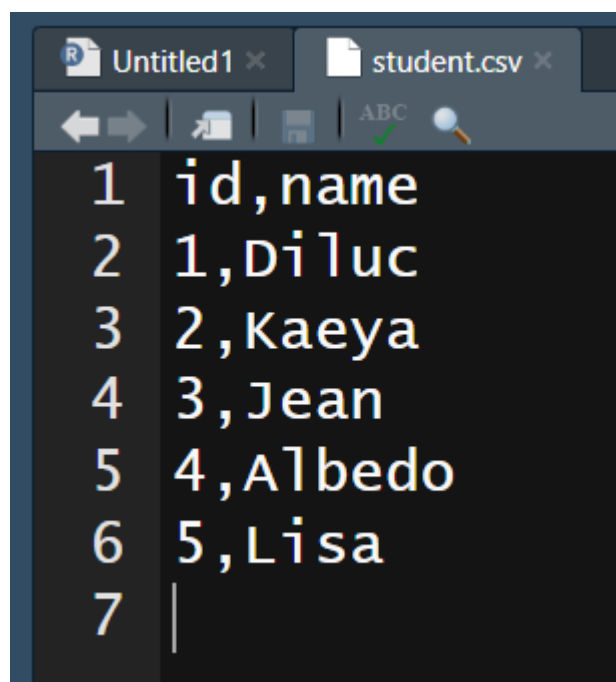


The screenshot shows an RStudio interface with a data frame named 'student' loaded. The data frame has two columns: 'id' and 'name'. The rows contain the following data:

	id	name
1	1	Diluc
2	2	Kaeya
3	3	Jean
4	4	Albedo
5	5	Lisa

Lesson 2: CSV

-ไฟล์ประเภท CSV (Comma-Separated Values) คือไฟล์ที่ใช้ comma เป็น delimiter คั่นระหว่าง column เช่นไฟล์ที่เราทำงานใน Google Sheets หรือ Excel ซึ่งเราสามารถใช้ R อ่านไฟล์ประเภทนี้ได้ด้วย `read_csv` เช่น:



The screenshot shows a text editor window with the content of a CSV file named 'student.csv'. The content is as follows:

```

1 id,name
2 1,Diluc
3 2,Kaeya
4 3,Jean
5 4,Albedo
6 5,Lisa
7 |

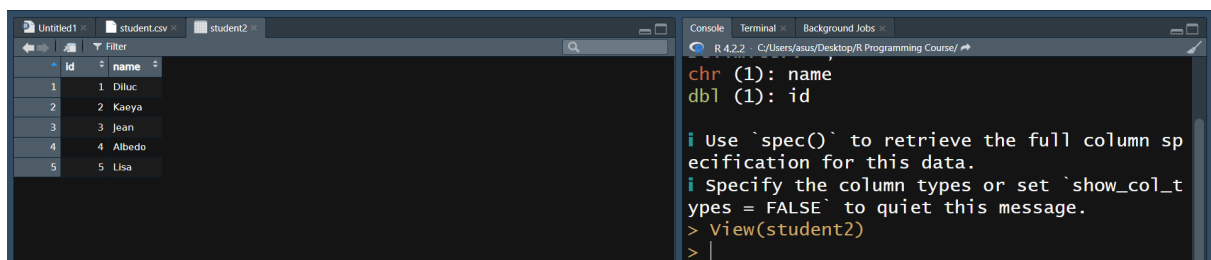
```

[ตัวอย่างของ .csv file]

```
> student2 <- read_csv("student.csv")
Rows: 5 Columns: 2
— Column specification —
Delimiter: ","
chr (1): name
dbl (1): id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_t
```

[read_csv ใช้อ่าน file ประเภท .csv ซึ่งเราสามารถ assign ผลกับตัวแปรเพื่อเรียกใช้งานต่อไปได้]



id	name
1	Diluc
2	Kaeya
3	Jean
4	Albedo
5	Lisa

```
chr (1): name
dbl (1): id

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_t
types = FALSE` to quiet this message.
> View(student2)
> |
```

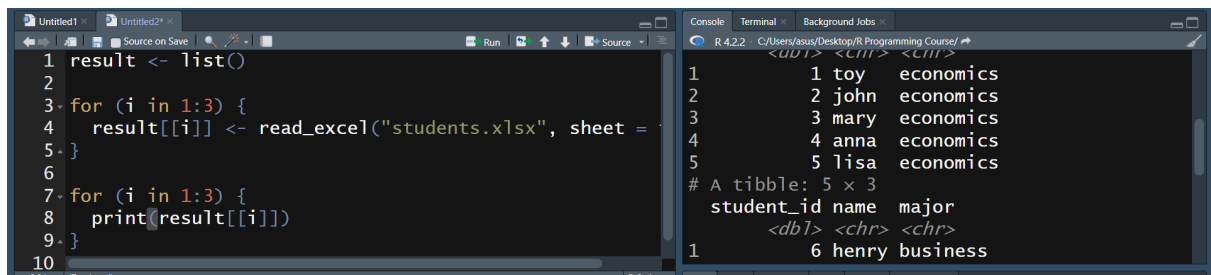
[สังเกตว่าหลังจาก View แล้ว จะแสดงผลเป็น Table เหมือนกัน]

Lesson3 : Excel File

-เราสามารถใช read_excel() จาก library ชื่อ readxl เพื่ออ่านไฟล์ประเภท .xlsx ได้ เช่น:

```
> library("readxl")
> read_excel("students.xlsx", sheet = 1)
# A tibble: 5 × 3
  student_id name major
  <dbl> <chr> <chr>
1         1 toy economics
2         2 john economics
3         3 mary economics
```

[sheet = 1 หมายความว่า อ่าน sheet แผ่นที่ 1 ใน file.xlsx. ไม่ได้หลาย sheet ซึ่งเราสามารถพิมพ์ sheet = "ชื่อของ sheet นั้น ๆ" เพื่อดู sheet ที่ต้องการได้เช่นกัน]



```
1 result <- list()
2
3 for (i in 1:3) {
4   result[[i]] <- read_excel("students.xlsx", sheet = i)
5 }
6
7 for (i in 1:3) {
8   print(result[[i]])
9 }
10
```

```
1      1 toy      economics
2      2 john    economics
3      3 mary    economics
4      4 anna    economics
5      5 lisa    economics
# A tibble: 5 x 3
#   student_id name      major
#   <dbl> <chr> <chr>
1      6 henry    business
```

```
#for loop to assign each sheet to result variable
for (i in 1:3) {
  result[[i]] <- read_excel("students.xlsx", sheet = i)
}

#for loop to print all sheets
for (i in 1:3) {
  print(result[[i]])
}
```

Lesson 4: Google Sheets

-read_sheet() ใช้อ่านข้อมูลจาก Google Sheets

-ถ้าเป็น public link ก่อนใช้คำสั่ง read_sheet() ให้รัน gs4_deauth() ก่อน (แปลว่า link ที่เราจะอ่าน file เป็น public link ไม่ต้องมีการ login)

-อ่านข้อมูลเพิ่มเติมเกี่ยวกับ gs4deauth() ได้ที่:

<https://googlesheets4.tidyverse.org/articles/auth.html>

-ตัวอย่าง code ในการอ่านข้อมูลจาก Google Sheets:

```
#Run the library necessary to use read_sheet()
library(google sheets4)
#Assign Google Sheets url to a variable
url <- "Google Sheets Link"
#If the file is open to public, use gs4_deauth
gs4_deauth()
```



```
#Then, use read_sheet to read Google Sheets's spreadsheet
read_sheet(url, sheet = "sheet's Name")
```

Lesson 5: JSON

-JSON = JavaScript Object Notation

-เราสามารถใช fromJSON() เพื่ออ่าน file ประเภท JSON เข้ามาเป็น List ใน R แล้วค่อยใช้ data.frame() เพื่อเปลี่ยน List → Data Frame ได้

-Key ใน JSON ต้องเขียนด้วย " " (Double Quote) เท่านั้น

-อ่านเพิ่มเติมเกี่ยวกับ jsonlite ได้ที่: <https://cran.r-project.org/web/packages/jsonlite/index.html>

```
{
  "id" : [1,2,3,4,5] ,
  "name" : ["Diluc","Kaeya","Jean","Albedo","Lisa"] ,
  "tail" : [true,true,true,false,true]
}
```

	id	name	tail
1	1	Diluc	TRUE
2	2	Kaeya	TRUE
3	3	Jean	TRUE
4	4	Albedo	FALSE
5	5	Lisa	TRUE

```
> teyvat2 <- data.frame(fromJSON("teyvat2.json"))
> View(teyvat2)
```

[ใช้ fromJSON("ชื่อไฟล์ JSON") เพื่ออ่านไฟล์ JSON จากนั้น ทำเป็น Data Frame แล้วเก็บค่าไว้ในตัวแปรเพื่อเรียกดูหรือใช้งานต่อไปได้]

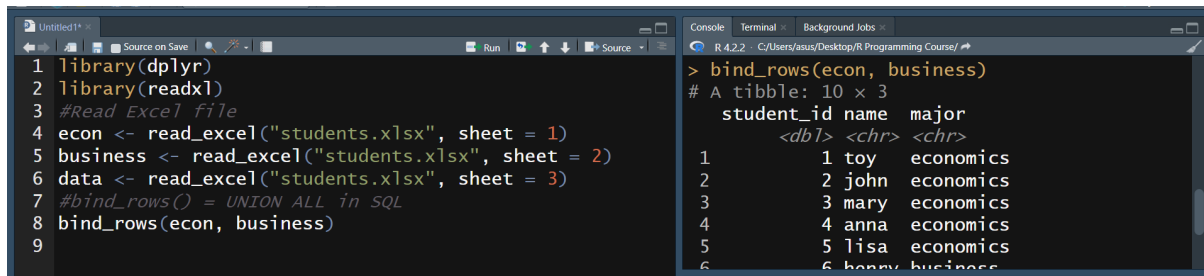
Lesson 6: Bind Rows (UNION ALL)

-bind_rows() ใน R = UNION ALL ใน SQL

```
library(dplyr)
library(readxl)
```

```
#Read Excel file
econ <- read_excel("students.xlsx", sheet = 1)
business <- read_excel("students.xlsx", sheet = 2)
data <- read_excel("students.xlsx", sheet = 3)
#bind_rows() = UNION ALL in SQL
```

-ตัวอย่างการใช้งาน:

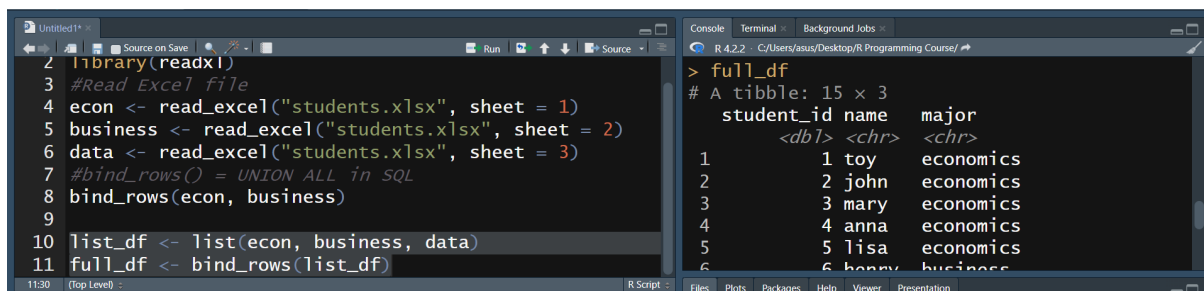


```
1 library(dplyr)
2 library(readxl)
3 #Read Excel file
4 econ <- read_excel("students.xlsx", sheet = 1)
5 business <- read_excel("students.xlsx", sheet = 2)
6 data <- read_excel("students.xlsx", sheet = 3)
7 #bind_rows() = UNION ALL in SQL
8 bind_rows(econ, business)
9
```

```
> bind_rows(econ, business)
# A tibble: 10 × 3
  student_id name      major
  <dbl>     <chr>   <chr>
1         1    toy    economics
2         2   john    economics
3         3   mary    economics
4         4   anna    economics
5         5   lisa    economics
6         6   henny business
```

-ในการทำงานจริง เราอาจมี Data Frame มากมาย ดังนั้น เราควรสร้าง List ของ Data Frame ขึ้นมาก่อน จากนั้นถึงค่อยใช้ bind_rows(list_df) เช่น:

```
list_df <- list(econ, business, data)
full_df <- bind_rows(list_df)
```



```
2 library(readxl)
3 #Read Excel file
4 econ <- read_excel("students.xlsx", sheet = 1)
5 business <- read_excel("students.xlsx", sheet = 2)
6 data <- read_excel("students.xlsx", sheet = 3)
7 #bind_rows() = UNION ALL in SQL
8 bind_rows(econ, business)
9
10 list_df <- list(econ, business, data)
11 full_df <- bind_rows(list_df)
```

```
> full_df
# A tibble: 15 × 3
  student_id name      major
  <dbl>     <chr>   <chr>
1         1    toy    economics
2         2   john    economics
3         3   mary    economics
4         4   anna    economics
5         5   lisa    economics
6         6   henny business
```

Lesson 7: Bind Cols (≠ JOIN)

-Bind Cols ไม่เท่ากับการเขียน JOIN เพราะไม่จำเป็นต้องใช้ Key ใด ๆ เพียงเอา Data Frame มาต่อข้างกันเท่านั้น

-rep() ช่วยในการทำซ้ำในกรณีที่มีข้อมูลเหมือนกัน เช่น

```
city = c("Mondstadt", "Mondstadt", "Mondstadt")
#Equals to:
city = c( rep("Mondstadt", 3) )
```

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

```
12
13 df1 <- data.frame(
14   id = 1:3,
15   name = c("Amber", "Lisa", "Kaeya")
16 )
17 df2 <- data.frame(
18   city = c( rep("Mondstadt", 3) ),
19   country = c( rep("Teyvat", 3) )
20 )
21
```

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

```
id name
1 1 Amber
2 2 Lisa
3 3 Kaeya
> df2
      city country
1 Mondstadt Teyvat
2 Mondstadt Teyvat
3 Mondstadt Teyvat
```

[เราจะทำการ Bind Cols 2 Data Frame นี้ด้วยกัน]

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

```
15 name = c("Amber", "Lisa", "Kaeya")
16 )
17 df2 <- data.frame(
18   city = c( rep("Mondstadt", 3) ),
19   country = c( rep("Teyvat", 3) )
20 )
21
22 bind_cols(df1, df2)
23 bind_cols(df2, df1)
24
```

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

```
> bind_cols(df1, df2)
  id name      city country
1 1 Amber Mondstadt Teyvat
2 2 Lisa Mondstadt Teyvat
3 3 Kaeya Mondstadt Teyvat
> bind_cols(df2, df1)
      city country id name
1 Mondstadt Teyvat 1 Amber
2 Mondstadt Teyvat 2 Lisa
```

[สังเกตว่า bind_cols(df1, df2) จะให้ผลลัพธ์ไม่เหมือนกับ bind_cols(df2, df1)]

-R สามารถ JOIN ได้เหมือนกับ SQL เช่น:

```
#df1
df1 <- data.frame(
  id = 1:3,
  name = c("Amber", "Lisa", "Kaeya")
)
#df2
df2 <- data.frame(
  id = 1:3,
  city = c( rep("Mondstadt", 3) ),
  country = c( rep("Teyvat", 3) )
)
#LEFT JOIN by id
left_join(df1, df2, by = "id")
```

```
17 df2 <- data.frame(  
18   id = 1:3,  
19   city = c( rep("Mondstadt", 3) ),  
20   country = c( rep("Teyvat", 3) )  
21 )  
22  
23 bind_cols(df1, df2)  
24 bind_cols(df2, df1)  
25 left_join(df1, df2, by = "id")  
26  
+   city = c( rep("Mondstadt", 3) ),  
+   country = c( rep("Teyvat", 3) )  
+ )  
> left_join(df1, df2, by = "id")  
  id name      city country  
1  1 Amber Mondstadt  Teyvat  
2  2 Lisa Mondstadt  Teyvat  
3  3 Kaeya Mondstadt Teyvat  
>
```

-bind_cols() อาจไม่ได้ใช้บ่อยเท่า JOIN เพราะการเขียน JOIN สามารถใช้ match id ของ 2 Data Frames หรือมากกว่าได้ เช่น customerid เป็นต้น

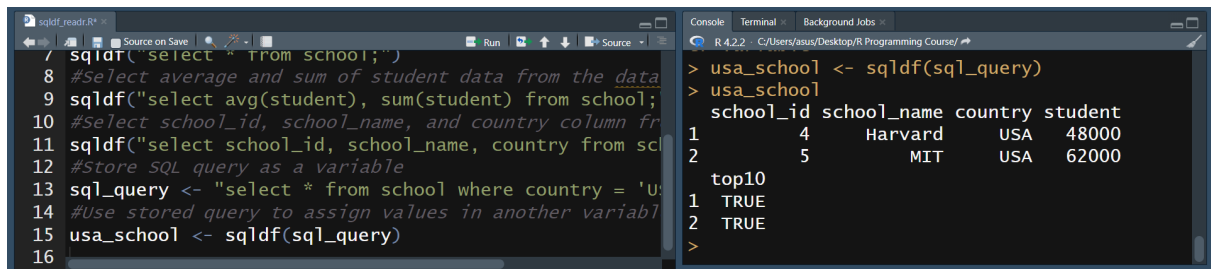
-ref. how to cross join in R?: <https://stackoverflow.com/questions/10600060/how-to-do-cross-join-in-r>

Lesson 8: SQL

-เราสามารถใช sqldf() เพื่อใช้เขียน SQL จัดการกับ Data Frame ใน R ได้ เช่น:

```
#Load library sqldf and readr  
library(sqldf)  
library(readr)  
  
#Assign school.csv to variable "school"  
school <- read_csv("school.csv")  
  
#Select all columns from school dataframe  
sqldf("select * from school;")  
  
#Select average and sum of student data from the dataframe (a  
sqldf("select avg(student), sum(student) from school;")  
  
#Select school_id, school_name, and country column from school  
sqldf("select school_id, school_name, country from school;")  
  
#Store SQL query as a variable  
sql_query <- "select * from school where country = 'USA';"
```

```
#Use stored query to assign values in variable "usa_school"
usa_school <- sqldf(sql_query)
```



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

```
8 sqldf("select * from school;")
9 #Select average and sum of student data from the data
9 sqldf("select avg(student), sum(student) from school;")
10 #Select school_id, school_name, and country column fr
11 sqldf("select school_id, school_name, country from scl
12 #Store SQL query as a variable
13 sql_query <- "select * from school where country = 'U
14 #Use stored query to assign values in another variabl
15 usa_school <- sqldf(sql_query)
16
```

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

```
> usa_school <- sqldf(sql_query)
> usa_school
  school_id school_name country student
1         4   Harvard    USA    48000
2         5     MIT      USA    62000
```

Below the table, the console shows the execution of `top10`:

```
top10
1 TRUE
2 TRUE
```

Lesson 9: SQLite

-เราสามารถ ใช้ library ชื่อ RSQLite เพื่อจัดการกับข้อมูลใน sqlite .db file

-ขั้นตอนการทำงานแบ่งเป็น 3 ขั้นตอน ได้แก่:

1. Connect to Database (Open Connection)
2. Get Data with SQL
3. Disconnect from Database (Close Connection)

-อ่านเพิ่มเติมเกี่ยวกับ Database ใน R ได้ที่: <https://dbi.r-dbi.org/>

-Code ตัวอย่างในการใช้ RSQLite:

```
#Load library
library(RSQLite)

#Connect to SQLite database (.db file)
#1. Open Connection
conn <- dbConnect(SQLite(), "chinook.db")

#2. Get Data
dbListTables(conn)
dbListFields(conn, "customers")

df <- dbGetQuery(conn, "select * from customers where country")
df2 <- dbGetQuery(conn, "select * from customers where country")
```

```
#3. Close Connection  
dbDisconnect(conn)
```

*อย่าลืมเช็คให้ดีๆ Database File ของเราอยู่ใน Working Directory แล้วหรือยัง

Lesson 10: How to save data in R

-เราสามารถ save data ใน R ได้ 2 แบบเพื่อนำกลับมาใช้ในอนาคต ต่อให้เราลบ data ทั้งหมด (ไม่มี data หลงเหลือบนหน้าต่าง environment แล้ว) ก็สามารถโหลดกลับมาใช้ใหม่ได้

1. `save.image()`: ใช้ save objects ทั้งหมดที่อยู่ใน environment (เช่น workspace) ของเรา เข้าไปที่ไฟล์ `.RData`
2. `saveRDS()`: ใช้ save single object แค่ไฟล์เดียวที่ไฟล์ `.rds`

-ถ้าต้องการโหลด `.RData` ให้ใช้ function `load()`

-ถ้าต้องการโหลด `.rds` ให้ใช้ function `readRDS()`

ยกตัวอย่างเช่น:

```
#save.image() + load()  
save.image(file = "data.RData")  
load(file = "data.RData")  
  
#saveRDS() + readRDS()  
saveRDS(business, file = "business.rds")  
business <- readRDS("business.rds")
```