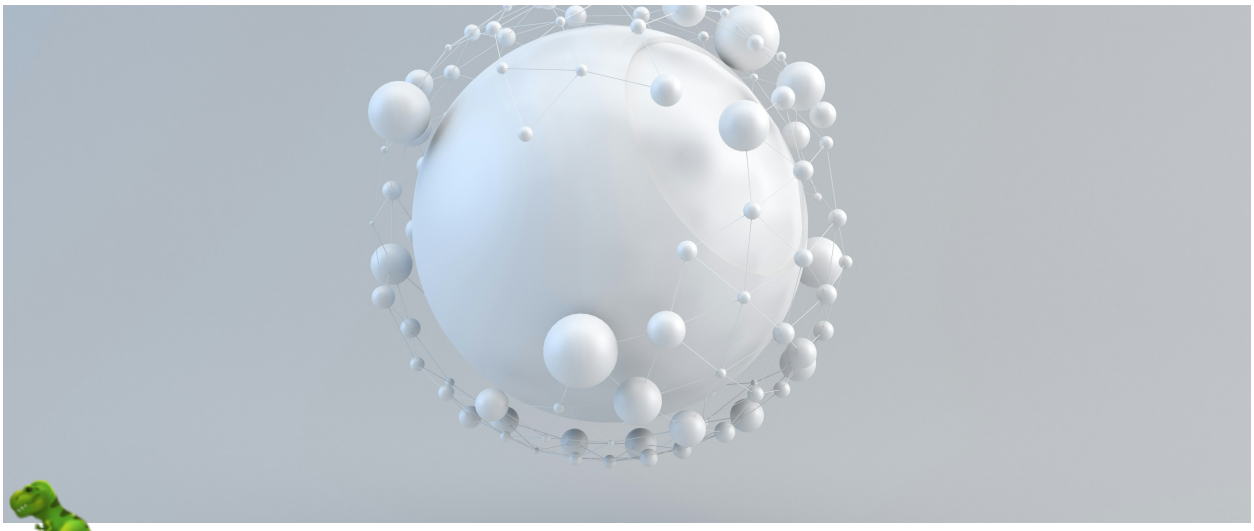

Data Transformation

Command line ที่พิมพ์ใน Terminal (Linux only)

```
# พิมพ์ข้อความออกมาเทียบเท่ากับคำสั่ง print ใช้คำสั่ง echo
/cloud/project$ echo "hello world" #hello world

# หาว่าเราอยู่โฟล์เดอร์ไหน
/cloud/project$ pwd #/cloud/project

# ดูว่ามีชื่อโฟลเดอร์อะไรบ้าง ตามด้วยชื่อ folder ก็ได้
/cloud/project$ ls #project.Rproj

# ถ้าให้โชว์ทุกไฟล์ด้วย รวม hidden fils ให้ใช้ สังเกตจะมี . ข้างหน้า
/cloud/project$ ls -a #.  ..  .Rhistory  .Rproj.user  project.Rp

# คำสั่งในการสร้างไฟล์ ใช้ touch ตามด้วยชื่อไฟล์
/cloud/project$ touch animal.txt

# และถ้าจะส่งค่าๆหนึ่งเข้าไปยังไฟล์นั้นให้ใช้ echo ตามด้วยค่านั้น
```

```
/cloud/project$ echo dog > animal.txt

# และถ้าจะเรียกดูว่าค่านั้นคืออะไรให้ใช้คำสั่ง cat ตามด้วยชื่อไฟล์
/cloud/project$ cat animal.txt

# อันนี้สำคัญ ถ้าเราใช้คำสั่งส่งค่าเข้าไปโดยใช้ echo dog > animal.txt ค่าใหม่จะเข้าไป
# และถ้าต้องการเพิ่มค่าเข้าไปเรื่อยๆให้ใช้ >>
/cloud/project$ echo cat >> animal.txt

# ถ้าต้องการลบไฟล์ให้ใช้ rm ถ้าเป็น rm -r ตามด้วยชื่อ folder จะเป็นการลบทุกไฟล์ที่อ
/cloud/project$ rm animal.txt
/cloud/project$ rm -r movie
/cloud/project$ rmdir movie #กรณีไม่มีไฟล์ใน folder

# ถ้่าต้องการสั่ง exicute code ในไฟล์นั้นๆให้ใช้คำสั่ง Rscript ตามโดยชื่อไฟล์
/cloud/project$ Rscript animal.txt

# การสร้าง folder ใหม่ให้ใช้คำสั่ง mkdir ตามด้วยชื่อ folder
/cloud/project$ mkdir movie

# และถ้าอยากพาตัวเองให้เข้าไปใน folder นั้นให้ใช้คำสั่ง cd ตามด้วยชื่อ folder
/cloud/project$ cd movie

# และถ้าอยากจะย้อน กลับไป folder ก่อนหน้าให้ใช้ cd .. และถ้าอยากย้อนกลับไปสองส
/cloud/project/movie$ cd ..
/cloud/project/movie$ cd ../..
```

Window Terminal

```
> echo hello world
# ปริ้นข้อความ

> echo %DATE%
# แสดงวันที่
```

```
> echo %TIME%
# แสดงเวลา

> set my_name = top # เก็บค่าไว้ในตัวแปร
> echo %my_name% # top

> cls # clear

> cd Desktop # ย้าย direcly ไปที่ Desktop
> mkdir Conicle # สร้าง folder
> notepad # เรียกใช้งาน notepad
> dir # เรียกดูไฟล์ทั้งหมด
> type name file # เรียกดู content ที่อยู่ในไฟล์

> cd .. # การกลับไป folder ก่อนหน้า ถ้าย้อนกลับ 2 step ..\..

> echo hello > hello.txt # สร้างไฟล์ใหม่ คำว่า hello ไปยังไฟล์
> echo this course is good >> hello.txt # เพิ่มบรรทัดใหม่ในไฟล์

> help dir # ดูรายละเอียด dir
> del = ชื่อไฟล์.txt # ลบไฟล์ หรือ * ลบไฟล์ทั้งหมด
> rmdir ชื่อ folder # ลบ folder
> rmdir /S ชื่อ folder #บังคับลบ folder กรณี folder มีไฟล์อยู่
> ren ชื่อเก่า ชื่อใหม่ # เปลี่ยนชื่อได้ทั้งไฟล์ และ folder

> notepad city.txt # สร้าง notepad เร้วๆ
> findstr /N "o" city.txt # หาตัว o ในไฟล์ว่ามีคำอะไรบ้างและอยู่แถวไหน

> curl แปะ URL --output ตั้งชื่อไฟล์.jpg
> curl แปะ URL --output ชื่อไฟล์.csv

> ping www.google.com # เช็คว่าเข้าถึงเว็บไซต์ได้ไหม active อยู่หรือป่าว

> move ชื่อไฟล์ที่ต้องการย้าย โฟลเดอร์ที่ต้องการเก็บไฟล์นั้น
> move ชื่อโฟลเดอร์\ไฟล์ที่อยู่ในโฟลเดอร์ . # ย้ายไฟโฟลเดอร์ curent direcly
```

Essentials Library

- tidyverse
- glue
- RSQlite
- RPostgreSQL
- lubridate

```
library(tidyverse)
library(glue)
library(lubridate)
```

glue message

การใช้งานคำสั่งฟังก์ชัน glue คือมันจะใช้งานง่ายกว่าคำสั่ง paste ในการ build string template

```
my_name <- "top"
my_age <- 21
glue("Hi my name is {my_name}. Today I'm {my_age} year old.")

# Hi my name is top. Today I'm 21 year old.
```

Data Transformation using dplyr(tidyverse)

- select
- filter
- arrange
- mutate ⇒ create new column
- summarize

** เราจะใช้ data mtcars สามารถเรียกใช้โดย พิมพ์ `view(mtcars)` ใน console

** สามารถเรียกดูชื่อคอลัมน์ทั้งหมดได้โดย `colnames()` และชื่อ index row โดย `rownames()`

** การเปลี่ยน index row ให้เป็น column และตั้งชื่อใหม่ได้ ให้ใช้ฟังก์ชัน `ชื่อเดต้าเฟรม <- rownames_to_column(ชื่อเดต้าเฟรม, "ชื่อคอลัมน์ใหม่")`

** ดูข้อมูลเบื้องต้นโดยใช้ `glimpse()` หรือ `head()` `tail()` เรียกดูชื่อคอลัมน์ใช้ `names()`

1. select() สามารถทำได้หลายแบบมาดูกันเลย

```
# การเลือกคอลัมน์แค่บางคอลัมน์สามารถเรียกด้วยตำแหน่งของคอลัมน์ เช่น 1 2 3 ... ห
# ชื่อคอลัมน์ก็ได้ถ้าเรารู้ชื่อเต็ม
select(mtcars, 1:5, model, mpg)

# ต่อมาคือการหาคอลัมน์ที่ขึ้นต้นด้วย... ลงท้ายด้วย... หรือว่าตัวอักษร... ที่อยู่ในชื่อคอ
select(mtcars, starts_with("a"))
select(mtcars, ends_with("p"))
select(mtcars, contains("a"))

# ส่วนมากจะเขียนแบบนี้มากกว่า โดยใช้ pipeline
mtcars %>%
  select(mpg, hp, wt) %>%
  head(5)

# เราสามรถเปลี่ยนชื่อคอลัมน์ที่เลือกมาได้
select(imdb, ชื่อใหม่ = ชื่อเก่า, ....)
```

2. filter()

```
# การใช้คำสั่ง filter สามารถ filter ได้หลายเงื่อนไขโดย และ(&), หรือ(|)
filter(mtcars, hp > 200 & mpg > 15)
filter(mtcars, hp > 200 | mpg > 15)

# ใช้ pipeline จะดูเท่กว่า
```

```r
mtcars %>%
  filter(hp > 200 & mpg >= 15)

# ใช้ร่วมกับ select
mtcars %>%
  select(mpg, hp, wt) %>%
  filter(hp >200)

# regular expression ใน r ฟังก์ชันคือ grep ดึงตำแหน่ง index หรือดึงค่าออกม
# ถ้าเป็น grepl จะได้ค่า TRUE, FALSE
grep("^M", mtcars$model) #จงหา model ที่ขึ้นต้นด้วย M ในตาราง mtcars
grep("^M", mtcars$model, value=T)
grepl("^M", mtcars$model) #จะได้ค่า TRUE FALSE

# ตัวอย่างเพิ่ม
imdb %>%
  select(movie_name, genre, rating) %>%
  filter(grepl("Drama", imdb$genre))     # ดึงแถวที่มี Drama มาหมด ม

mtcars %>%
  select(model, mpg, hp, wt, am) %>%
  filter(grepl("^D", model)) # คำตอบที่ได้คือตารางที่มี model ขึ้นต้นด้วยตัว
# filter มันจะดึงมาเฉพาะค่า TRUE และแสดงผลเป็นค่า

# การ filter ระหว่างค่า (between)
mtcars %>%
  select(model, mpg, hp, wt, am) %>%
  filter(between(hp, 150, 200))

# กรณีมีการใช้ หรือ เยอะเราจะใช้ in operator เข้ามาช่วย
imdb %>%
  select(movie_name, length, score) %>%
  filter(score %in% c(8.3, 9.0))
```

3. arrange() คือการ sort data นั้นแหละ จากมากไปน้อย จากน้อยไปมาก

```
# เราอยากจะ sort คอลัมน์ไหน ก็ใส่คอลัมน์นั้น defeult มันจะเรียงจากน้อยไปมาก
arrange(mtcars, hp)
arrange(mtcars, desc(hp))

# สามารถ sort ได้หลายคอลัมน์
mtcars %>%
  select(model, mpg, hp, wt, am) %>%
  filter(between(hp, 150, 200)) %>%
  arrange(am, desc(hp))
```

😊 # write csv files `write_csv(ชื่อเดต้าเฟรม หรือชื่อตัวแปร, "result.csv", row.names = FALSE)`

4. mutate() คือฟังก์ชันในการสร้างคอลัมน์ใหม่ ซึ่งมันจะสร้างออกมาทางด้านขวาเสมอ

```
# หลักการสร้างคือ mutate(ชื่อคอลัมน์ใหม่ = สมการ) ถ้าชื่อคอลัมน์ใหม่เป็นคอลัมน์เก่ามันจ
mtcars %>%
  select(model, mpg, hp, wt, am) %>%
  filter(mpg >= 20) %>%
  mutate(model_upper = toupper(model), #เปลี่ยนคอลัมน์ model ทั้งหมดเป็
         mpg_double = mpg*2,           #เอา mpg มาคูณสอง
         mpg_hahaha = mpg_double+10,   #สามารถเอาคอลัมน์ที่พึ่งสร้างใหม่เมื่
         am_label = if_else(am==0, "auto", "mannual")) #ใช้คำสั่ง i
```

5. summarize() คือมันจะคล้ายๆกับ aggregate ใน sql

```
# หลักการคือ summarize(ชื่อตัวแปรที่ต้องการประกาศ = ฟังชันก์)
mtcars %>%
  summarize(avg_mpg = mean(mpg),
            med_mpg = median(mpg),
```

```
            sum_mpg = sum(mpg),
            min_mpg = min(mpg),
            max_hp = max(hp),
            var_hp = var(hp),
            sd_hp = sd(hp),
            n = n()) #นับจำนวนแถว หรือจะใช้ nrow() แทนก้ได้

#avg_mpg med_mpg sum_mpg min_mpg max_hp   var_hp
#1 20.09062    19.2   642.9    10.4    335 4700.867
#     sd_hp  n
#1 68.56287 32

# group by คือ เราสามารถสั่งกลุ๊ปคอลัมน์นั้นก่อนที่ summarize ข้อมูลได้
mtcars %>%
  mutate(am = if_else(am==0, "auto", "mannual")) %>%
  group_by(am) %>%
  summarize(avg_mpg = mean(mpg),
            sum_mpg = sum(mpg),
            min_mpg = min(mpg),
            max_hp = max(hp),
            n = n())

#am      avg_mpg sum_mpg min_mpg max_hp     n
# <chr>     <dbl>   <dbl>   <dbl>  <dbl> <int>
#1 auto       17.1    326.    10.4    245    19
#2 mannual    24.4    317.    15      335    13
```

Join DataFrame

```
# การ join ตารางเข้าด้วยกันมีฟังก์ชันง่ายๆ ดังนี้
left_join(ชื่อเดต้าเฟรมแรก, ชื่อเดต้าเฟรมที่สอง, by="ชื่อคอลัมน์ที่เชื่อมกัน")

left_join(band_members, band_instruments, by="name")
```

```
inner_join(band_members, band_instruments, by="name")
full_join(band_members, band_instruments, by="name")

# สามารถเขียนใน pipeline ได้
band_members %>%
     mutate(name_upper = toupper(name)) %>%
     left_join(band_instruments, by="name")

# ในกรณีที่ Key ของเดต้าเฟรมทั้งสองตารางไม่ตรงกัน
band_members %>%
  select(member_name = name,   #สามารถเปลี่ยนชื่อคอลัมน์ได้ในคำสั่ง select
         band_name = band) %>%
  left_join(band_instruments,
            by = c("member_name" = "name")) #เมื่อชื่อ key ไม่ตรงกันก็ต้อ
```

Tibble vs. Data.frame

```
library(tidyverse)

# datafram vs. tibble

tibble(id = 1:3, name = c("toy", "jisoo", "liza")) # print แค่ 10
data.frame(id = 1:3, name = c("toy", "jisoo", "liza"))

# convert dataframe to tibble
mtcars_tibble <- tibble(mtcars)
mtcars_tibble

# ใน r ส่วนมากจะใช้ tibble เพราะการแสดงผลสวยกว่า

# การ slice() เพื่อดึง rows ที่ต้องการ
mtcars %>%
  slice(1:5)

mtcars %>%
```

```
  slice(c(1, 3, 5))

mtcars %>%
  slice(sample(nrow(mtcars), 10 )) # ผลลัพธ์เหมือนกับ sample_n
```

```
> tibble(id = 1:3, name = c("toy", "jisoo", "liza"))
# A tibble: 3 × 2
     id name
  <int> <chr>
1     1 toy
2     2 jisoo
3     3 liza
> data.frame(id = 1:3, name = c("toy", "jisoo", "liza"))
  id  name
1  1   toy
2  2 jisoo
3  3  liza
```

Sampling

```
# สามารถเลือกสุ่มแถวตามจำนวนที่เราต้องการได้
mtcars %>%
  sample_n(2) %>%  #เลือกสุ่มมาสองแถว
  pull(model)      #แสดงคำตอบเป็น vector ไม่ใช่ตาราง เพื่อนำเอาไปใช้ต่อ

# หรือสามารถเรียกสุ่มออกมาเป็น % ตามที่เราต้องการก็ได้
mtcars %>%
  sample_frac(0.2, replace = TRUE) %>% #สุ่มออกมา 20% และ replace =
  pull(model)

# อันนี้สำคัญ การที่เราสุ่มออกมาเพื่อนำข้อมูลไปใช้ต่อเราเรียกว่า Statistics ซึ่งในการทำง
mtcars %>%
```

```
  sample_frac(0.2) %>%
  summarize(avg_hp = mean(hp)) #ค่าเฉลี่ยที่ได้จะใกล้เคียงกับข้อมูลที่ไม่ได้สุ่มตัวอย่
```

Count

```
# การใช้คำสั่ง count คือการนับภายในคอลัมน์คล้ายๆ groupby แต่ถ้าเป็นการนับแนะนำให้
mtcars %>%
  count(am)

#am  n
#1 Auto 19
#1 Manual 13

# เราก็สามารถเอา n ไปใช้ต่อได้ เช่นสร้างคอลัมน์คิดเป็น % ออกมาสิว่า Auto คิดเป็นกี่ %
mtcars %>%
  count(am) %>%
  mutate(pct = n/sum(n)) #n มาจาก คอลัมน์ count

#am  n pct
#1 Auto 19 0.59375
#2 Manual 13 0.40625
```

การเขียน sql

```
# ก่อนอื่นต้องโหลด packages ของ spl ก่อนและเปิด library ขึ้นมา
install.packages("sqldf")
library(sqldf)

# สามารถเขียนได้ดังตัวอย่างด้านล่างเลย
sqldf("select mpg, hp from mtcars
      where hp >= 200"
```

```
)

sqldf("select avg(hp), max(hp), count(*)
      from mtcars"
)
```

การเชื่อมต่อเข้ากับ Database

```
library(RSQLite) # ก่อนอื่นต้องโหลด library และเชื่อมต่อไปยัง server ที่เราใช้

con <- dbConnect(SQLite(), "chinook.db")  # connect ไปยังเดต้าเบส

dbListTables(con) # เรียกดูชื่อตารางทั้งหมด
dbListFields(con, "customers") # เรียกดูชื่อคอลัมน์ของตารางนั้นๆ

dbGetQuery(con, "select firstname, email from customers # เขียนเรี
                    where country = 'USA'")

# create dataframe
products <- tribble(
  ~id, ~product_name,
  1, "chocolate",
  2, "pineapple",
  3, "sumsung"
)
# ถ้าต้องการดูการแสดงผลแบบ datafram ให้ใช้ฟังก์ชัน ซึ่งการแสดงผล tribble แสด
as.data.frame(products)

# อันนี้สำคัญคือการนำตารางที่เราสร้างมาเมื่อกี้ ยัดเข้าไปใน เดต้าเบสของเรา
dbWriteTable(con, "products", products) # ชื่อเดต้าเบสที่ connect, ชื่อต

# ต่อมาถ้าต้องการลบตารางในเดต้าเบส
dbRemoveTable(con, "products")
```

```
# และทุกครั้งที่เลิกใช้งาน ต้องปิด connection ด้วย
dbDisconnect(con)
```

Building Database by ElephantSQL

```
# ก่อนอื่น Create Database บน ElephantSQL ก่อน
# หลังจากนั้นเขียนคำสั่งตามโค้ดด้านล่างเพื่อเชื่อมต่อกัน
library(RPostgreSQL)
library(tidyverse)

con <- dbConnect(
  PostgreSQL(),
  host = "floppy.db.elephantsql.com",
  dbname = "bjyengyo",
  user = "bjyengyo",
  password = "bp7QkbM-ZZBf9tAP4STs4kQlpJbzqPi1",
  port = 5432
)
#เมื่อเชื่อมต่อแล้ว พิมพ์ con จะขึ้นว่า <PostgreSQLConnection>
#จากนั้นก็สามารถเขียนฟังก์ชันต่างๆที่เรียนมา และส่งขึ้นไปยัง Postgre ได้เลย
```

Homework