



Code

▼ Data Transformation

```
## install.packages
## install.packages("dplyr")

## load packages
library(readr)
library(dplyr)

## read csv file into RStudio
imdb <- read.csv("imdb.csv", stringsAsFactors = FALSE)
View(imdb)

## review data structure
glimpse(imdb)

## print head and tail of data
```

```
head(imdb, 10) # ดูข้อมูล 10 แถวบนสุด
tail(imdb, 10) # ดูข้อมูล 10 แถวล่างสุด
```

Note : stringsAsFactor = False เป็นการบอกว่าให้มองว่าข้อมูลที่เป็นตัวอักษรจากไฟล์ csv เป็น character

Select column

```
## select columns
select(imdb, MOVIE_NAME, RATING)
select(imdb, 1, 2)

## rename columns
select(imdb, movie_name = MOVIE_NAME, released_year = YEAR)
#select(Table_name, new_name = old_name)

## pipe operator
imdb %>%
  select(movie_name = MOVIE_NAME, released_year = YEAR) %>%
  head(10)
```

`dplyr` มาพร้อมกับ pipe operator `%>%` ที่ใช้ในการเขียน pipeline ตัวอย่าง code ด้านล่าง ทั้งสองแบบได้ผลลัพธ์เหมือนกัน

```
# load dplyr
library(dplyr)

# select columns
select(mtcars, mpg, wt, hp)
mtcars %>% select(mpg, wt, hp)
```

ข้อดีของการเขียน `%>%` คือเราสามารถเชื่อมฟังก์ชันของเราได้หลาย steps เช่น

```
df %>% select() %>% filter() %>% mutate() %>% arrange()
```

Filter Data

```
## filter data
filter(imdb, SCORE >= 9.0)
imdb %>% filter(SCORE >= 9.0)

names(imdb) <- tolower(names(imdb))

imdb %>%
  select(movie_name, year, score) %>%
  filter(score >= 9.0 & year > 2000) # and operator

imdb %>%
  select(movie_name, length, score) %>%
  filter(score == 8.8 | score == 8.3 | score == 9.0) # or operator

imdb %>%
  select(movie_name, length, score) %>%
  filter(score %in% c(8.8, 8.3, 9.0)) # in operator
```

```
## filter string columns
imdb %>%
  select(movie_name, genre, rating) %>%
  filter(grepl("Drama", imdb$genre)) # grepl : pattern matching

imdb %>%
  select(movie_name) %>%
  filter(grepl("The", imdb$movie_name)) # grepl is a case sensitive
```

Create New Columns

```
## create new columns
imdb %>%
  select(movie_name, score, length) %>%
  mutate(score_group = if_else(score >= 9, "High Rating", "Low Rating"))
```

```

length_group = if_else(length >= 120, "Long Film", "

imdb %>%
  select(movie_name, score) %>%
  mutate(score = score + 0.2) %>%
  head(10)

```

Arrange Data

```

## arrange data
head(imdb)

imdb %>%
  arrange(length) %>%
  head(10)

imdb %>%
  arrange(desc(length)) %>% # descending order
  head(10)

imdb %>%
  arrange(rating, desc(length))

```

Summary Statistics

```

## summarise and group_by
imdb %>%
  filter(rating != "") %>%
  group_by(rating) %>%
  summarise(mean_length = mean(length),
            sum_length = sum(length),
            sd_length = sd(length),
            min_length = min(length),

```

```
max_length = max(length),  
n = n()) # n() การนับจำนวน row ทั้งหมดใน dataframe
```

Join Data

```
## join data  
favorite_films <- data.frame(id = c(3,14,25,46,58))  
  
favorite_films %>%  
  inner_join(imdb, by = c("id" = "no"))  
#inner_join(name_table_refer, by = c("foreign_key" = "primary
```

Export Data (csv file)

```
## write csv file (export result)  
imdb_prep <- imdb %>%  
  select(movie_name, released_year = year, rating, length, sc  
  filter(rating == "R" & released_year > 2000)  
  
## export file  
write.csv(imdb_prep, "imdb_prep.csv", row.names = FALSE)
```

▼ Intermediate Data Transformation

Tibble

🌱 `tibble()` vs. `data.frame()` ใน R เราชอบใช้ tibble มากกว่า data.frame แบบปกตินะ ครับ เพราะการแสดงผลจะสวยงามกว่า

Tip - ถ้าอยากจะ print tibble มากกว่า 10 แถว ให้ใช้ฟังก์ชัน `print(mtcars_tibble, n=20)` กำหนดจำนวนแถวที่จะ print ได้ที่ parameter `n`

```
# install.packages("tidyverse")
# dplyr tidyr ggplot2
library(tidyverse)


# data.frame vs. tibble

df_tibble <- tibble(id = 1:3, name = c("Folk", "Mameaw", "Soo
data.frame(id = 1:3, name = c("Folk", "Mameaw", "Soobin")))

# convert dataframe to tibble
mtcars
mtcars_tibble <- tibble(mtcars)
# tibble have data type and fit output
```

Note : tibble จะมี datatype ของแต่ละ column และแสดงผลหน้าตาต่างไปตามขนาดของหน้าจอ

Sample Data

 `sample_n()` สุ่มจำนวน rows ตามที่เราต้องการ หรือ `sample_frac()` สุ่มเป็น % ได้ i.e. frac ย่อมาจากคำว่า fraction

```
## sample_n

set.seed(24)
sample_n(mtcars, size = 8)

sample_frac(mtcars, size = 0.5, replace = T) # replace = TRUE
```

Note : run function `set.seed()` และ `sample_n()` พร้อมกันจะเป็นการสุ่มตัวอย่างเดิม(พืชตัวอย่าง)

Slice

 `slice()` ใช้ดึง rows ที่เราต้องการจาก dataframe / tibble

```
## slice
mtcars %>%
  slice(1:5)

mtcars %>%
  slice(6:10)

mtcars %>%
  slice(c(2,4,6))

mtcars %>%
  slice(sample(nrow(mtcars),10))
# note : slice() ใช้ดึง rows ที่เราต้องการจาก dataframe / tibble
```