BÀI THỰC HÀNH 4: THAO TÁC VÀ LÀM SẠCH DỮ LIỆU ĐÃ THU THẬP

1. Toán tử %>%

Toán tử **%>%** trong R được gọi là toán tử luồng (tạm dịch, tiếng Anh: pipe operator). Toán tử này được dùng để liên kết chuỗi các hàm lại với nhau để xử lý cho một thao tác nhất định trên dữ liệu (ví dụ: select - filter - select).

Toán tử này do **Stefan Milton Bache** tạo nên, và được hỗ trợ trong R bởi thư viện *magrittr*. Để cài đặt thư viện trên, ta dùng lệnh

```
install.packages('magrittr').
```

Để sử dụng toán tử trên, ta phải import thư viện magrittr vào R bằng lệnh:

```
library('magrittr')
```

Một số ví dụ về sử dụng toán tử %>%

Truy xuất thông thường	Truy xuất bằng toán tử %>%
filter(data, variable == *numeric_value*)	data %>% filter(variable == *numeric_value*)
a <- filter(data, variable == *numeric_value*) b <- summarise(a, Total = sum(variable)) c <- arrange(b, desc(Total))	data %>% filter(variable == "value") %>% summarise(Total = sum(variable)) %>% arrange(desc(Total))

Các thao tác ở phần sau sẽ áp dụng toán tử %>% này để tiện truy xuất.

2. Thao tác dữ liệu với DPLYR

Thư viện DPLYR là một thư viện hỗ trợ thao tác với dữ liệu một cách dễ dàng và có cấu trúc hơn so với việc thao tác theo cách truyền thống. Thư viện DPLYR hỗ trợ các thao tác chính như sau:

1. mutate(): thêm biến (thuộc tính) vào dữ liệu hiện tại.
2. select(): lấy dữ liệu theo biến (thuộc tính).
3. filter(): lọc giá trị của dữ liệu theo điều kiện.
4. summarise(): thống kê dữ liệu theo thuộc tính.

5: group_by(): Gom nhóm dữ liệu theo thuộc tính.

5. arrange(): sắp xếp dữ liệu.

Với các thao tác như trên, việc truy xuất với các dữ liệu sẽ dễ dàng hơn cho người dùng.

Thư viện dpylr được cung cấp tại CRAN ở địa chỉ:

https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

Để cài đặt thư viện trên trong R Studio, ta dùng lệnh:

`install.packages('dplyr').`

Để sử dụng thư viện, ta dùng lệnh: `library('dplyr').`

Bộ dữ liệu: **Novel Corona Virus 2019 dataset**

Link: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

Đọc dữ liệu và lưu vào biến data. Sử dụng lệnh: `read.csv().`

```
data <- read.csv("corona_virus/covid_19_data.csv")
data$ObservationDate <- as.Date(data$ObservationDate, tz =
"UTC", "%Y/%m/%d")
```

Thao tác select trên dữ liệu:

1. Lấy dữ liệu ở các thuộc tính bao gồm: ngày quan sát (ObservationDate), quốc gia nhiễm (Country.Region), số ca dương tính (Confirmed), số ca tử vong (Deaths).

```
data %>% select(ObservationDate, Country.Region,
Confirmed, Deaths)
```

2. Lấy dữ liệu về số ca hồi phục ở từng quốc gia, thuộc tính hiển thị gồm: ngày quan sát (ObservationDate), quốc gia nhiễm (Country.Region) và số ca hồi phục (Recovered).

```
data %>% select(ObservationDate, Country.Region,
Recovered)
```

3. Tính tổng số ca dương tính: sử dụng hàm sum(), thuộc tính: Confirmed.

```
sum(data %>% select(Confirmed))
```

4. Tương tự câu 2, nhưng lấy ra 10 dòng đầu tiên:

```
head(data %>% select(ObservationDate, Country.Region,
Confirmed, Deaths), 10)
```

Thao tác lọc dữ liệu filter:

Với thao tác này, việc lấy dữ liệu ra theo điều kiện giá trị dễ hơn so với sử dụng lệnh **which()** ở các bài trước.

1. Lấy dữ liệu về số ca nhiễm ở Trung Quốc (Mainland China):

```
data %>% filter(Country.Region == "Mainland China")
```

2. Lấy dữ liệu về số ca nhiễm của Việt Nam trong tháng 03 và tháng 04.

```
data %>% filter(Country.Region == "Mainland China" &
ObservationDate >= "2020-03-01" & ObservationDate <=
"2020-04-30")
```

Thao tác thống kê dữ liệu với summarise:

Thống kê dữ liệu về số ca dương tính của China: số ca dương tính trung bình, trung vị, phương sai và độ lệch chuẩn.

```
data %>% filter(Country.Region == "Mainland China") %>%
        summarise(
            Mean=mean(Confirmed, na.rm = TRUE),
            Median = median(Confirmed, na.rm = TRUE),
            Variance = var(Confirmed, na.rm = TRUE),
            SD = sd(Confirmed, na.rm = TRUE)
        )
```

Thao tác gom nhóm dữ liệu với group_by:

Hiển thị dữ liệu theo từng ngày quan sát (thuộc tính ObservationDate) của Việt Nam trong 2 tháng: tháng 3 và tháng 4 năm 2020.

```
data %>% filter(
                Country.Region == "Vietnam" &
                  ObservationDate >= "2020-03-01" &
                  ObservationDate <= "2020-04-30") %>%
        group_by(ObservationDate)
```

Thao tác sắp xếp dữ liệu với arrange:

1. Hiển thị dữ liệu theo từng ngày quan sát (thuộc tính ObservationDate) của Việt Nam trong 2 tháng: tháng 3 và tháng 4 năm 2020. Sắp xếp theo số ca dương tính tăng dần.

```
data %>% filter(
                Country.Region == "Vietnam" &
                  ObservationDate >= "2020-03-01" &
                  ObservationDate <= "2020-04-30") %>%
         group_by(ObservationDate) %>%
         arrange(Confirmed)
```

2. Tương tự như trên, nhưng sắp xếp số ca dương tính giảm dần. Để sắp xếp giảm dần một thuộc tính, ta dùng thao tác: ***desc(<thuộc_tính>)*** trong hàm arrange.

```
data %>% filter(
                Country.Region == "Vietnam" &
                  ObservationDate >= "2020-03-01" &
                  ObservationDate <= "2020-04-30") %>%
         group_by(ObservationDate) %>%
         arrange(desc(Confirmed))
```

<u>Thêm vào một thuộc tính mới sử dụng mutate:</u>

Hiển thị dữ liệu theo từng ngày quan sát (thuộc tính ObservationDate) của Việt Nam trong 2 tháng: tháng 3 và tháng 4 năm 2020. Sắp xếp theo số ca dương tính giảm dần. Thêm thuộc tính Patients = số lượng ca dương tính (Confirmed) - số ca phục hồi (Recovered).

```
data %>% filter(
                Country.Region == "Vietnam" &
                ObservationDate >= "2020-03-01" &
                ObservationDate <= "2020-04-30") %>%
        group_by(ObservationDate) %>%
        arrange(desc(Confirmed)) %>%
        mutate(Patient = Confirmed - Recovered)
```

Ngoài ra, còn một số thao tác khác như: join, merge, ...

3. Làm sạch dữ liệu với TIDYR

Thư viện TIDYR là một trong các thư viện hỗ trợ cho việc làm sạch dữ liệu, tạo ra các dữ liệu đáp ứng yêu cầu của **Tidy data** theo Hadley Wickham.

Thư viện ***tidyr*** này có sẵn trên CRAN tại địa chỉ:

https://cran.r-project.org/web/packages/tidyr/index.html

Để cài đặt thư viện tidyr trong R, ta sử dụng lệnh:

`install.packages('tidyr').`

Import thư viện tidyr trong R bằng cách: `library('tidyr').`

Các thao tác làm sạch dữ liệu trong tidyr:

1. gather(): gom dữ liệu theo từng cột ứng với các cặp key-values tương ứng.
2. spread(): chia dữ liệu ra thành từng cột.
3. separate(): chia cột ra thành nhiều cột.
4: unite(): gom nhiều cột lại thành 1 cột.

Bộ dữ liệu: **MTCars**

Link: https://www.kaggle.com/ruiromanini/mtcars

Đọc dữ liệu trong R bằng hàm read.csv() và lưu vô biến *data*:

```
data <- read.csv("mtcars/mtcars.csv")
```

Bộ dữ liệu này gồm 32 điểm dữ liệu và 12 thuộc tính:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
12	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
13	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
15	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
16	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
17	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
18	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1

Thao tác gom dữ liệu với gather:

Gom dữ liệu các thuộc tính (ứng với các cột), trừ thuộc tính model lại thành 2 thuộc tính: ***attribute*** và ***value*** tương ứng. Giá trị lưu trong biến *gathered*.

```
gathered <- data %>% gather(attribute, value, -model)
```

Kết quả:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
12	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
13	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
15	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
16	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
17	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4

	model	attribute	value
1	Mazda RX4	mpg	21.000
2	Mazda RX4 Wag	mpg	21.000
3	Datsun 710	mpg	22.800
4	Hornet 4 Drive	mpg	21.400
5	Hornet Sportabout	mpg	18.700
6	Valiant	mpg	18.100
7	Duster 360	mpg	14.300
8	Merc 240D	mpg	24.400
9	Merc 230	mpg	22.800
10	Merc 280	mpg	19.200
11	Merc 280C	mpg	17.800
12	Merc 450SE	mpg	16.400
13	Merc 450SL	mpg	17.300
14	Merc 450SLC	mpg	15.200
15	Cadillac Fleetwood	mpg	10.400
16	Lincoln Continental	mpg	10.400

Trước khi gom thuộc tính (12) Sau khi gom thuộc tính (3)

Nhận xét: số cột sau khi sử dụng hàm gather() sẽ giảm đi so với ban đầu, số dòng sẽ tăng lên.

Chia dữ liệu ở 1 thuộc tính ra thành từng cột với hàm spread.

Đây là hàm ngược lại so với hàm gather. Hàm này sẽ tạo ra các cột mới ứng với các thuộc tính và dữ liệu ứng với value.

Chia dữ liệu ở cột ***attribute*** ra thành các cột, giá trị ở từng cột ứng với giá trị ở ***value***. Giá trị thu được lưu vào biến `spread`.

spread <- gathered %>% spread(attribute, value)

Nhận xét: số cột sau khi sử dụng hàm spread sẽ nhiều hơn, số dòng sẽ ít lại.

Để minh hoạ cho hai hàm: unite và separate, ta tạo data ảo (fake) như sau, và lưu vào biến *`data`*.

```
set.seed(1)
date <- as.Date('2016-01-01') + 0:14
hour <- sample(1:24, 15)
min <- sample(1:60, 15)
second <- sample(1:60, 15)
event <- sample(letters, 15)
data <- data.frame(date, hour, min, second, event)
```

	date	hour	min	second	event
1	2016-01-01	7	30	29	u
2	2016-01-02	9	43	36	a
3	2016-01-03	13	58	60	l
4	2016-01-04	20	22	11	q
5	2016-01-05	5	44	47	p
6	2016-01-06	18	52	37	k
7	2016-01-07	19	12	43	r
8	2016-01-08	12	35	6	i
9	2016-01-09	11	7	38	e
10	2016-01-10	1	14	21	b
11	2016-01-11	3	20	42	w
12	2016-01-12	14	1	32	t
13	2016-01-13	23	19	52	h
14	2016-01-14	21	41	26	s
15	2016-01-15	8	16	25	o

Dữ liệu sau khi tạo

Gom dữ liệu ở nhiều cột lại thành một cột sử dụng hàm unite.

Gom dữ liệu ở các cột date, hour, min, second lại thành một thuộc tính là datetime. Dữ liệu thu được lưu vào biến `fullTime`.

Ta thực hiện 2 bước:

Bước 1: Gom các thuộc tính: *date*, *hour* lại thành **datehour**, 2 thuộc tính nối với nhau bởi khoảng trắng.

Bước 2: Gom các thuộc tính: *datehour*, *min*, *second* lại thành thuộc tính **datetime**, các thuộc tính nối với nhau bởi dấu hai chấm ":".

```
fullTime <- data %>%
    unite(datehour, date, hour, sep = ' ') %>%
```

```
    unite(datetime, datehour, min, second, sep = ':')
```

Kết quả sau khi thực hiện

	date	hour	min	second	event
1	2016-01-01	7	30	29	u
2	2016-01-02	9	43	36	a
3	2016-01-03	13	58	60	l
4	2016-01-04	20	22	11	q
5	2016-01-05	5	44	47	p
6	2016-01-06	18	52	37	k
7	2016-01-07	19	12	43	r
8	2016-01-08	12	35	6	i
9	2016-01-09	11	7	38	e
10	2016-01-10	1	14	21	b
11	2016-01-11	3	20	42	w
12	2016-01-12	14	1	32	t
13	2016-01-13	23	19	52	h
14	2016-01-14	21	41	26	s
15	2016-01-15	8	16	25	o

Trước khi gom

	datetime	event
1	2016-01-01 7:30:29	u
2	2016-01-02 9:43:36	a
3	2016-01-03 13:58:60	l
4	2016-01-04 20:22:11	q
5	2016-01-05 5:44:47	p
6	2016-01-06 18:52:37	k
7	2016-01-07 19:12:43	r
8	2016-01-08 12:35:6	i
9	2016-01-09 11:7:38	e
10	2016-01-10 1:14:21	b
11	2016-01-11 3:20:42	w
12	2016-01-12 14:1:32	t
13	2016-01-13 23:19:52	h
14	2016-01-14 21:41:26	s
15	2016-01-15 8:16:25	o

Sau khi gom

Tách dữ liệu ra ở một cột thành nhiều cột sử dụng hàm separate

Hàm **separate** ngược lại so với hàm **gather**. Hàm này sẽ tách dữ liệu ở một cột ra thành nhiều cột theo các ký tự đặc tả.

Với dữ liệu *fullTime*, tách dữ liệu ở cột **datetime** thành các cột: *date*, *hour*, *min*, *second*.

Các bước tách dữ liệu:

Bước 1: Tách dữ liệu ở cột ***datetime*** thành 2 cột: *date*, *time* dựa theo ký tự khoảng trắng.

Bước 2: Tách dữ liệu ở cột ***time*** thành các cột: *hour*, *min*, *second* dựa theo ký tự dấu hai chấm ":".

```
fullTime %>%
    separate(datetime, c('date', 'time'), sep = ' ') %>%
    separate(time, c('hour', 'min', 'second'), sep = ':')
```

4. Thao tác với ngày tháng sử dụng thư viện lubridate

Thư viện lubridate là một thư viện được thiết kế để thao tác dễ dàng với kiểu dữ liệu ngày tháng (datetime trong R). Các thao tác bao gồm: định dạng ngày tháng, tính toán trên ngày tháng và truy xuất dữ liệu với định dạng ngày tháng (vd: lấy tháng, năm, ngày, thứ, ...).

Thư viện trong lubridate có sẵn trên CRAN ở địa chỉ:

https://cran.r-project.org/web/packages/lubridate/vignettes/lubridate.html

Để cài đặt thư viện lubridate, ngoài thư viện lubridate ta còn phải cài thêm thư viện tidyverse.

```
install.packages('tidyverse')
install.packages('lubridate')
```

Để import thư viện **lubridate**, ta dùng lệnh: `library('lubridate')`.

Bộ dữ liệu: **Novel Corona Virus 2019 dataset**

Link: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

Đọc dữ liệu, và lưu vào biến data.

data <- read.csv("corona_virus/covid_19_data.csv")

Chuyển dữ liệu của ObservationDate thành kiểu Date trong R

data$ObservationDate <- as.Date(data$ObservationDate, tz = "UTC", "%m/%d/%Y")

1. Lấy năm nhuận: dùng hàm `leap_year(<ngày_tháng>)`.

```
print(data[1,])
leap_year(data$ObservationDate[1])
```

2. Lấy năm: dùng `year()`

```
print(data[1,])
year(data$ObservationDate[1])
```

3. Lấy tháng: dùng `month()`

```
print(data[1,])
month(data$ObservationDate[1])
```

4. Lấy ngày: Dùng `day()`

```
print(data[1,])
day(data$ObservationDate[1])
```

5. Định dạng ngày tháng: dùng để format ngày tháng in ra.

ymd(): năm-tháng-ngày.

ymd_hms(): năm-tháng-ngày giờ-phút-giây.

mdy(): tháng-ngày-năm.

dmy(): ngày-tháng-năm.

hms(): giờ-phút-giây.

6. Lấy giờ hiện tại: `now()`.

7. Lấy ngày hiện tại: `today()`.

8. Lấy thứ hiện tại: `wday(time, label=TRUE, abbr=FALSE)`.

label: tên thứ trong tuần:

TRUE: lấy tên thứ trong tuần. VD: Monday, Tuesday, ...,

FALSE: lấy thứ trong tuần theo số. Bắt đầu từ Sunday là số 1.

abbr: tên thứ viết tắt hay viết thường.

TRUE: viết đầy đủ. VD: Monday, Tuesday, ...

FASLE: viết tắt. VD: Mon, Tue, Wed, ...

9. Lấy khoảng thời gian tiếp theo: `make_difftime(<thời gian>)`

<*thời gian*>: day, month, year, hour, second, hour, min.

Ghi chú: Số âm là lấy ngày trước đó.

VD: Lấy khoảng thời gian 5 ngày tiếp theo.

```
make_difftime(day=5)
```

Ví dụ 1: Lấy dữ liệu về số ca nhiễm dương tính từ tháng 01 đến tháng 03 sử dụng lubridate:

```
data %>% filter(Country.Region == "Vietnam" &
month(ObservationDate) %in% c(1,3))
```

Ví dụ 2: Lấy dữ liệu về số ca nhiễm dương tính trong tất cả các ngày thứ 4.

```
data %>% filter(Country.Region == "Vietnam" &
wday(ObservationDate, label=TRUE) == "Wed")
```

5. Bài tập

Sử dụng lại bộ dữ liệu *Novel Corona 2019 Dataset* ở trên.

Bài 1:

a) Các bạn hiện thực lại các ví dụ ở trên.

b) Tìm dữ liệu về số ca nhiễm của Nhật Bản từ ngày 02/3/2021 đến ngày 15/03/2021. Vẽ biểu số ca nhiễm theo từng ngày. (sử dụng hàm plot)

c) Tìm dữ liệu về số ca nhiễm của Hoa Kỳ từ ngày 15/03/2021 đến ngày 15/04/2021. Vẽ biểu đồ số ca nhiễm theo từng ngày (biểu đồ đường - hàm *plot*) và vẽ biểu đồ ***đếm số ca nhiễm được ghi nhận*** theo từng bang (biểu đồ cột - hàm *barplot*).

Bài 2: Thống kê số ca nhiễm mới của thế giới theo từng ngày, từ tháng 02 đến tháng 04 của năm 2020. Vẽ biểu đồ số ca nhiễm theo từng ngày (biểu đồ đường).

Gợi ý: Dùng group by gom nhóm và tính tổng số ca nhiễm theo từng ngày quan sát.

Bài 3: Thống kê số ca nhiễm mới của Việt Nam theo từng tháng trong năm 2021. Vẽ biểu đồ số ca nhiễm theo từng tháng (biểu đồ đường).

Gợi ý: Tách thuộc tính **ObservationDate** ra thành 3 phần: ngày / tháng / năm. Sau đó gom nhóm (group by) theo từng tháng và tính tổng số ca nhiễm mới.

Bài 4: Thống kê số ca nhiễm mới của Việt nam theo từng thứ trong tháng 04 năm 2020.

Gợi ý: sử dụng group by theo hàm **wday** (hàm **wday** thuộc về thư viện lubridate).

Bài 5: Hãy thống kê số ca nhiễm mới tại Việt Nam trong khoảng tháng **01 - 03/2020** và tháng **01 - 03/2021,** sử dụng ***tứ phân vị*** (dùng hàm *quantile*)

Bài 6: *Vẽ biểu đồ so sánh **tổng số ca nhiễm mới** của Việt nam giữa tháng 04 năm 2019, tháng 04 năm 2020 và tháng 04 năm 2021.

Bài 7: *Vẽ biểu đồ boxplot, so sánh số ca nhiễm tại Việt Nam trong khoảng tháng **01 - 03/2020** và tháng **01 - 03/2021.**

Nộp bài: Các bạn nộp file source code và file báo cáo đi kèm.

Quy định nộp bài: Nộp 2 file code và báo cáo, không nén file, đặt tên theo cú pháp:

<MSSV>_<Họ tên>_BT4.pdf

và

<MSSV>_<Họ tên>_BT4.R

Chúc tất cả các bạn học tốt

PHỤ LỤC: TÍNH TOÁN ĐỘ CHÊNH LỆCH GIỮA DỮ LIỆU HIỆN TẠI VÀ DỮ LIỆU TRƯỚC ĐÓ HOẶC DỮ LIỆU KẾ TIẾP

Để tìm dữ liệu trước dữ liệu hiện tại, ta dùng hàm **lag()** trong thư viện **dplyr**
Để tìm dữ liệu sau dữ liệu hiện tại, ta dùng hàm **lead()** trong thư viện **dplyr**

Ví dụ: Dữ liệu trong bộ covid19 như sau:

ObservationDate	Confirmed
15/04/2020	233
16/04/2020	240
17/04/2020	250
18/04/2020	257

Để tìm số ca nhiễm mới tăng lên, ta lấy số ca nhiễm của ngày hiện tại trừ cho số ca nhiễm của ngày trước đó

```
data_covid_19 %>% mutate(NewCase = Confirmed -
lag(Confirmed, default = first(Confirmed))
```

Kết quả thư được như sau:

ObservationDate	Confirmed	NewCase
15/04/2020	233	NA
16/04/2020	240	13
17/04/2020	250	10
18/04/2020	257	7