Bài giảng R, số 1
-Khám phá phân tích dữ liệu với R-

TS.Tô Đức Khánh

26/02/2024

1 Giới thiệu

Trong bài này, chúng ta sẽ học về các bước khám phá phân tích dữ liệu (Exploratory Data Analysis) cho hai loại dữ liệu có cấu trúc, tức là, dữ liệu định lượng (numerical data/quantitative data) và dữ liệu định tính (categorical data/quanlitative data), với phần mềm `R`. Cụ thể hơn, ta sẽ học cách sử dụng các thư viện trong `R`, cho các nhiệm vụ sau:

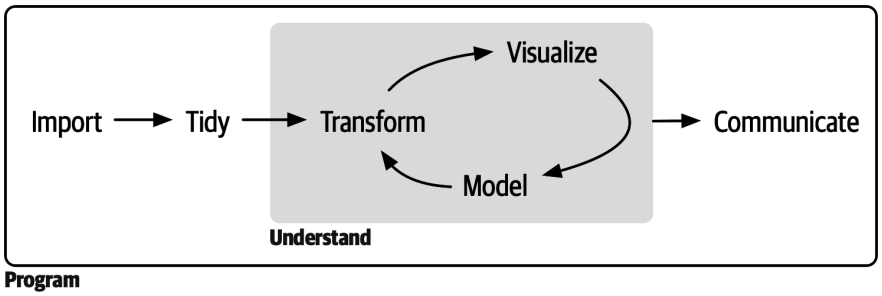
- nhập dữ liệu
- làm sạch dữ liệu
- ước lượng các giá trị đặc trưng của dữ liệu (location, variability)
- ước lượng phân phối của dữ liệu
- sự tương quan
- khám phá dữ liệu định tính
- khám phá hai hoặc nhiều biến
- biểu diễn đa biến

Các thư viện chính bao gồm:

- `tidyverse`
- `ggplot2`

Tài liệu chính của bài học này, tới từ cuốn sách "R for data science: Import, Tidy, Transform, Visualize and Model data" của nhóm tác giả Hadley Wickham, Mine Çetinkaya-Rundel và Garrett Grolemund. Bản online của cuốn sách có thể truy cập theo địa chỉ sau: https://r4ds.hadley.nz/

Một quy trình đọc và xử lý số liệu thống kê được mô tả như trong Hình 1.



Hình 1: Quy trình phân tích xử lý số liệu.

Cụ thể:

1. *import* - đọc/nhập dữ liệu từ tệp tin lưu trữ (có thể là tệp .txt, hoặc .csv, hay .xls) vào trong không gian làm việc của `R`;

2. *tidy* - sắp xếp dữ liệu lại sao cho gọn gàng nhằm dễ quản lý và truy xuất thông tin biến, đối tượng, chú ý, một dữ liệu gọn gàng (một cách ngắn gọn) là một dữ liệu mà mỗi một cột sẽ chứa thông tin của một biến, và mỗi dòng sẽ là thông tin của một đối tượng quan sát;
3. *transform* - Một khi dữ liệu đã được nhập và sắp xếp gọn gàng, ta có thể tạo thêm các biến mới (bằng cách áp dụng các phép biến đổi, logarithm, căn bậc hai, ...) hoặc tính các thống kê tổng hợp: trung bình, tần số, tỷ số ...
4. *visualize* - Mô tả dữ liệu (hay thống kê mô tả) là một quan trọng, giúp ta có một cái nhìn tổng quát về dữ liệu, nhằm phát hiện ra các xu hướng (có thể là đã được dự đoán hoặc chưa được dự đoán từ trước), qua đó giúp ta có định hướng được các phân tích tiếp theo nhằm giải quyết câu hỏi nghiên cứu ban đầu. Phần mô tả này có thể được hoàn thành bởi việc sử dụng bảng tổng hợp, hoặc các biểu đồ.
5. *model* - Một khi ta đã xác định rõ câu hỏi nghiên cứu, ta sẽ cần các mô hình thống kê để tìm ra câu trả lời. Mô hình ở đây có thể là mô hình kiểm định hay mô hình hồi quy. Chú ý rằng, mọi mô hình đều yêu cầu những giả định nhất định, ví dụ, giả định về phân phối chuẩn. Do đó, ta nhất thiết phải kiểm tra những giả định này. Một khi giả định không thỏa, mô hình hiện tại cần được thay đổi, hoặc ta có thể xem xét tới các phép biến đổi dữ liệu để thỏa được giả định.
6. *communicate* - Bước cuối cùng trong công đoạn phân tích dữ liệu là trao đổi, đánh giá kết quả thu được sau quá trình phân tích.

2 Nhập dữ liệu và kiểm soát dữ liệu từ một tệp .csv

Trong phần này, ta tập trung vào rectangular data.

Trong phần này, ta sẽ làm quen với cách đọc một tệp dữ liệu đuôi .csv bằng hàm `read_csv()` trong thư viện `readr` một trong những thư viện được đính kèm trong thư viện `tidyverse`.

```
library(tidyverse)
```

Cấu trúc cơ bản của một đoạn code `R` đọc tệp dữ liệu sẽ gồm hai phần, như sau:

```
ten_data_trong_R <- read_csv("duong_dan_toi_tep_du_lieu")
```

- phần bên trái dấu "`<-`" là tên của dữ liệu được lưu lại trong không gian làm việc của `R`;
- phần bên phải dấu "`<-`" là cú pháp để nhập dữ liệu với đường dẫn cụ thể tới tệp dữ liệu, chú ý, đường dẫn này có thể là một đường liên kết tới một địa chỉ online;
- dấu "`<-`" là toán tử gán giá trị vào trong một biến.

Ví dụ, ta xét bộ dữ liệu `students.csv` về thực phẩm và chế độ ăn của 7 học sinh.

```
data_students <- read_csv("datasets/students.csv")
```

```
## Rows: 6 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Full Name, favourite.food, mealPlan, AGE
## dbl (1): Student 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 message.
```

Để nhìn thấy bảng dữ liệu mới được nhập vào, ta có thể gõ trực tiếp tên của biến lưu dữ liệu, trong trường hợp này là `data_students`, hoặc có thể dùng cú pháp `View(data_students)`. Ví dụ:

```
data_students
```

1	Student ID	Full Name	favourite.food	mealPlan	AGE
2	1	Sunil Huffmann	Strawberry yoghurt	Lunch only	4
3	2	Barclay Lynn	French fries	Lunch only	5
4	3	Jayendra Lyne	N/A	Breakfast and lunch	7
5	4	Leon Rossini	Anchovies	Lunch only	
6	5	Chidiegwu Dunkel	Pizza	Breakfast and lunch	five
7	6	Güvenç Attila	Ice cream	Lunch only	6

Hình 2: Dữ liệu về thực phẩm và chế độ ăn của 7 học sinh.

```
## # A tibble: 6 x 5
##   `Student ID` `Full Name`      favourite.food     mealPlan            AGE
##          <dbl> <chr>            <chr>              <chr>               <chr>
## 1            1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
## 2            2 Barclay Lynn     French fries       Lunch only          5
## 3            3 Jayendra Lyne    N/A                Breakfast and lunch 7
## 4            4 Leon Rossini     Anchovies          Lunch only          <NA>
## 5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
## # i 1 more row
```

Từ kết quả, ta thấy rằng, dữ liệu gồm có 6 dòng (tức là 6 quan sát) và 5 cột (tương ứng với 5 biến), đã được nhập vào đúng như trong tệp .csv gốc, với tên của các cột được hiển thị ở dòng trên của mỗi cột, theo sau là viết tắt của dạng biến, ở đây:

- `<dbl>` là viết tắt của "double", tức là dạng số (bao gồm cả số thực và số nguyên);
- `<chr>` là viết tắt của "character" tức là dạng chữ.

Ngoài ra, ta cũng có thể nhập dữ liệu từ một địa chỉ online:

```
data_students <- read_csv("https://pos.it/r4ds-students-csv")
```

Chú ý: Ngoài hàm `read_csv()` ta còn có các hàm để đọc tệp dữ liệu như sau:

- `read_xls()`, `read_xlsx()` hoặc `read_excel()` trong thư viện `readxl`, dùng để đọc tệp dữ liệu dạng tệp excel, đuôi .xls hoặc .xlsx;
- `read_table()` trong thư viện `readr` được dùng để đọc các tệp dữ liệu dạng bảng được lưu dưới dạng tệp có đuôi .txt hoặc .dat.

2.1 Kiểm soát tên và dạng dữ liệu

Sau khi đã đọc dữ liệu, bước đầu tiên thường liên quan đến việc chuyển đổi dữ liệu đó theo một cách nhất định để giúp ta làm việc dễ dàng hơn trong phần còn lại của phân tích.

2.1.1 Giá trị khuyết

Quan sát cột đồ ăn ưu thích `favourite.food`, ta dễ dàng nhận thấy một giá trị `N/A`, nó có thể là ký hiệu cho một giá trị bị khuyết (tức là không được ghi nhận lại). Trong `R`, giá trị bị khuyết của một biến sẽ được ký hiệu chuẩn là `NA`, tất cả các ký hiệu khác đều sẽ được coi là dạng chữ (nếu ký hiệu đó là chữ), và trong

trường hợp này là `N/A`. Để hiệu chỉnh lỗi dạng này, ta có thể hiệu chỉnh trực tiếp trên tệp dữ liệu, hoặc có thể hiệu chỉnh bởi đối số `na = ...` trong hàm `read_csv()`, tức là

```
ten_data_trong_R <- read_csv("duong_dan_toi_tep_du_lieu", na = c("", "NA"))
```

Mặc định, đối số `na = c("", "NA")` tức là sẽ chỉ định ô trống (`""`) hoặc `"NA"` là dữ liệu khuyết, do đó, ta chỉ cần cập nhật thêm trường hợp vào trong `na = ...`, tức là `na = c("", "NA", "N/A")`. Như vậy, đoạn đọc dữ liệu được sửa lại như sau:

```
data_students <- read_csv("datasets/students.csv", na = c("", "NA", "N/A"))
```

```
## Rows: 6 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Full Name, favourite.food, mealPlan, AGE
## dbl (1): Student 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 message.
```

và ta thu được

```
data_students
```

```
## # A tibble: 6 x 5
##   `Student ID` `Full Name`      favourite.food     mealPlan            AGE
##          <dbl> <chr>            <chr>              <chr>               <chr>
## 1            1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
## 2            2 Barclay Lynn     French fries       Lunch only          5
## 3            3 Jayendra Lyne    <NA>               Breakfast and lunch 7
## 4            4 Leon Rossini     Anchovies          Lunch only          <NA>
## 5            5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
## # i 1 more row
```

2.1.2 Tên của feature (biến)

Tên của các biến trong `R` phải được khai báo với chữ viết thường, khoảng cách giữa các chữ phải được thay thế bằng dấu gạch chân (tạo thành từ dạng hình con rắn - *snake case*), và không bắt đầu với chữ số. Ví dụ, `blood`, `short_flights`, `short4`.

Đối với bảng dữ liệu `data_students`, tên của biến đầu tiên `‘Student ID‘` là sai định dạng tiêu chuẩn, do có chữ cái viết hoa “S” và “I”, đồng thời, có khoảng cách giữa hai từ “Student” và “ID”. Tương tự, các tên biến còn lại cũng không đúng tiêu chuẩn. Để khắc phục điều này một cách tự động, ta có thể sử dụng hàm `clean_names()` trong thư viện `janitor`

```
library(janitor)
```

với cú pháp như sau:

```
data_students <- read_csv("datasets/students.csv", na = c("", "NA", "N/A")) |>
  clean_names()
```

```
## Rows: 6 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Full Name, favourite.food, mealPlan, AGE
## dbl (1): Student 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 message.
```

Dữ liệu mới có dạng như sau:

```
data_students
```

```
## # A tibble: 6 x 5
##   student_id full_name        favourite_food     meal_plan           age
##        <dbl> <chr>            <chr>              <chr>               <chr>
## 1          1 Sunil Huffmann   Strawberry yoghurt Lunch only          4
## 2          2 Barclay Lynn     French fries       Lunch only          5
## 3          3 Jayendra Lyne    <NA>               Breakfast and lunch 7
## 4          4 Leon Rossini     Anchovies          Lunch only          <NA>
## 5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch five
## # i 1 more row
```

Có thể thấy tên của các biến đã được chỉnh sửa phù hợp với tiêu chuẩn, một cách tự động.

Chú ý: cú pháp trên là sự kết hợp của hai đoạn lệnh, đoạn đầu tiên

```
read_csv("datasets/students.csv", na = c("", "NA", "N/A"))
```

dùng để đọc tệp dữ liệu, trong khi đó, đoạn thứ 2

```
clean_names()
```

dùng để điều chỉnh lại tên của các biến trong dữ liệu, kết quả sau đó được lưu lại vào tên biến trong không gian làm việc `R`. Sự kết hợp này được thực hiện thông qua toán tử `|>`, được gọi là "pipe", và sẽ được dùng phổ biến trong khâu đọc, chỉnh sửa và các thao tác khác trên bộ dữ liệu (lọc, tính toán các thống kê tổng hợp). Khi ta sử dụng `|>` có nghĩa là ta đang thực hiện ghép nối các kết quả từ bên trái sang phép toán bên phải, ví dụ, `x |> f(y)` sẽ tương đương với `f(x, y)`, và `x |> f(y) |> g(z)` sẽ tương đương là `f(x, y) |> g(z)` tức là `g(f(x, y), z)`.

Đọc thêm về pipe, trong chương 4 của sách "R for Data Science".

2.1.3 Kiểm soát dạng dữ liệu

Bước tiếp theo, ta cần phải kiểm soát các loại biến có mặt trong dữ liệu. Như đã nói ở phần trên, thể loại biến được thể hiện ở dòng tiếp theo tên biến, trong ví dụ này, ta có 1 biến với dạng số `<dbl>` và 4 biến dạng chữ `<chr>`. Trong số các biến dạng chữ, ta dễ nhận dàng nhận thấy rằng biến `age` đang bị định dạng sai kiểu, nó đáng ra phải là dạng số `<dbl>`. Quan sát kỹ hơn, ta nhận thấy có một giá trị `five` thay vì `5`. Như vậy, ta cần hiệu chỉnh là biến `age` mà không tác động tới tệp dữ liệu nguồn.

```
data_students <- data_students |>
  mutate(age = parse_number(if_else(age == "five", "5", age)))
```

Ở đoạn lệnh trên, ta dùng `mutate()` để tạo biến mới (cột mới) trong dữ liệu, ở đây, ta lấy tên biến mới cũng là `age`. Giá trị của biến mới này là giá trị của biến `age` cũ, ngoại trừ giá trị `five` được chuyển thành `5`, bằng cách sử dụng hàm `if_else()` (xem thêm trong chương 12, bản online của sách). Cuối cùng, để chuyển đổi `<chr>` sang `<dbl>`, ta có thể dùng hàm `parse_number()`. Kết quả dữ liệu mới

```
data_students
```

```
## # A tibble: 6 x 5
##   student_id full_name        favourite_food     meal_plan             age
##        <dbl> <chr>            <chr>              <chr>               <dbl>
## 1          1 Sunil Huffmann   Strawberry yoghurt Lunch only              4
## 2          2 Barclay Lynn     French fries       Lunch only              5
## 3          3 Jayendra Lyne    <NA>               Breakfast and lunch     7
## 4          4 Leon Rossini     Anchovies          Lunch only             NA
```

```
## 5          5 Chidiegwu Dunkel Pizza              Breakfast and lunch     5
## # i 1 more row
```

Một điểm cần lưu ý trước khi đi vào phân tích, đó là các biến dạng chữ `<chr>` mà có một tập hợp các giá trị có thể đã biết, cần phải được chuyển đổi sang dạng nhân tố (*factor*) `<fct>` nếu muốn sử dụng trong các phân tích thống kê (xem chương 16, bản online của sách, để biết thêm chi tiết về dạng nhân tố trong `R`). Để làm được điều này, ta có thể dùng hàm `factor()`, cụ thể như sau:

```
data_students <- data_students |>
  mutate(meal_plan = factor(meal_plan))
```

Mặc định, hàm `factor()` sẽ xếp thứ tự của cấp độ của một nhân tố theo thứ tự bảng chữ cái hoặc chữ số.

```
levels(data_students$meal_plan)
```

```
## [1] "Breakfast and lunch" "Lunch only"
```

Chú ý: Để truy xuất một cột biến trong bảng dữ liệu, ta dùng cách gọi `ten_du_lieu$ten_bien`.

Như vậy, tổng hợp tất cả các công đoạn phía trên, ta có đoạn lệnh hoàn chỉnh như sau:

```
data_students <- read_csv("datasets/students.csv", na = c("", "NA", "N/A"))
data_students <- data_students |>
  clean_names() |>
  mutate(age = parse_number(if_else(age == "five", "5", age)),
         meal_plan = factor(meal_plan))
```

2.2 Nhập dữ liệu từ nhiều file .csv

Giả sử ta có dữ liệu được lưu trữ trong nhiều file .csv khác nhau, mỗi file cùng chứa một lượng thông tin (số lượng features giống nhau), ta muốn nhập dữ liệu từ các file này trong không gian làm việc `R`, trong cùng 1 dữ liệu duy nhất. Để làm điều này, ta vẫn dùng

```
read_csv(list_path_file, id = "file")
```

với

- `list_path_file` là một vector chứa các đường dẫn của các tập dữ liệu cần đọc;
- đối số `id = "file"` tạo ra một cột mới có tên `file` vào trong bảng dữ liệu, nhằm chỉ rõ tên file dữ liệu gốc.

Ví dụ, ta muốn nhập dữ liệu từ 3 file .csv, lần lượt là `01-sales.csv`, `02-sales.csv` và `03-sales.csv`

```
sales_files <- c("datasets/01-sales.csv", "datasets/02-sales.csv",
                 "datasets/03-sales.csv")
data_sales <- read_csv(sales_files, id = "file")
```

```
## Rows: 19 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): month
## dbl (4): year, brand, item, n
##
## 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 message.
```

```
data_sales
```

```
## # A tibble: 19 x 6
##   file                  month    year brand  item     n
```

```
##   <chr>                <chr>   <dbl> <dbl> <dbl> <dbl>
## 1 datasets/01-sales.csv January  2019     1  1234     3
## 2 datasets/01-sales.csv January  2019     1  8721     9
## 3 datasets/01-sales.csv January  2019     1  1822     2
## 4 datasets/01-sales.csv January  2019     2  3333     1
## 5 datasets/01-sales.csv January  2019     2  2156     9
## # i 14 more rows
```

Sau đó, ta có thể tiến hành các công đoạn xử lý tiền dữ liệu, như kiểm soát tên, giá trị được ghi nhận, khuyết dữ liệu, dạng biến, như trường hợp nhập dữ liệu từ một file.

2.3 Các thao tác cơ bản trên dữ liệu

Trong phần này, chúng ta tìm hiểu cách sử dụng một số hàm được cung cấp của thư viện `dplyr`. Ta sẽ sử dụng dữ liệu `flights`, bao hàm thông tin các chuyến bay cất cánh từ thành phố New York trong năm 2013. Dữ liệu này được tổng hợp bởi US Bureau of Transportation Statistics, và được cung cấp trong thư viện `nycflights13`. Ta dùng hàm `data()` để nhập liệu được cung cấp trong một thư viện.

```
library(nycflights13)
data(flights)
flights
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## # i 336,771 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Theo thông tin được hiển thị, bảng dữ liệu có 336,776 dòng (ghi chép) và 19 cột (features). Để có được thông tin tổng hợp của 19 features này, ta có thể dùng hàm `glimpse()`:

```
glimpse(flights)
```

```
## Rows: 336,776
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1, 0, ~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849, 853,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851, 856,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -14, 31~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA", ~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 49, 71~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39463", ~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "JFK~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "MCO~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 158, 3~
```

```
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, 1028,~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, ~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0, 0, ~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 05:00:~
```

Để biết mô tả chi tiết của các biến trong bộ dữ liệu, hay tra cứu `help` về `flights`, hoặc

```
?flights
```

2.3.1 Lọc dữ liệu

Để lọc dòng dữ liệu theo điều kiện của một hoặc nhiều feature, ta sẽ sử dụng hàm `filter()`, với đối số là điều kiện cần lọc. Chú ý, ở đây, ta sẽ luôn sử dụng cấu trúc dòng lệnh dạng pipe:

```
data_name |> filter(dieu_kien_feature)
```

Để định nghĩa điều kiện lọc, ta cần một số định nghĩa cho các toán tử logic:

- `==`: so sánh bằng
- `!=`: khác
- `>`: lớn hơn
- `<`: bé hơn
- `>=`: lớn hơn hoặc bằng
- `<=`: khác

Ví dụ, ta muốn lọc ra các chuyến bay có thời gian chậm cất cánh (`dep_delay`) nhiều hơn 120 phút:

```
flights |> filter(dep_delay > 120)
```

```
## # A tibble: 9,723 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      848           1835       853     1001           1950
## 2  2013     1     1      957            733       144     1056            853
## 3  2013     1     1     1114            900       134     1447           1222
## 4  2013     1     1     1540           1338       122     2020           1825
## 5  2013     1     1     1815           1325       290     2120           1542
## # i 9,718 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Và nếu như chúng ta mong muốn lưu kết quả lọc này vào trong một biến, để tiện cho các phân tích sau này, ta có thể làm như sau:

```
flights_delay_120 <- flights |> filter(dep_delay > 120)
flights_delay_120
```

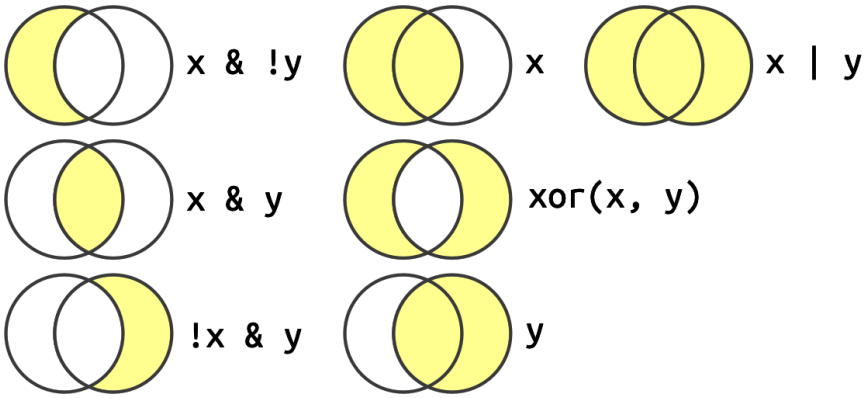
```
## # A tibble: 9,723 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      848           1835       853     1001           1950
## 2  2013     1     1      957            733       144     1056            853
## 3  2013     1     1     1114            900       134     1447           1222
## 4  2013     1     1     1540           1338       122     2020           1825
## 5  2013     1     1     1815           1325       290     2120           1542
```

```
## # i 9,718 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Ta có thể kết hợp nhiều điều kiện của nhiều features trong một điều kiện chung. Khi này, ta cần các toán tử để định nghĩa phép toán đại số Boolean (phép toán tập hợp). Trong `R`,

- `&` có nghĩa là "và"
- `|` có nghĩa là "hoặc"
- `!` có nghĩa là "không" (not)
- `xor()` có nghĩa là hiệu giữa phần hội và phần giao (exclusive or)

Chi tiết minh họa ở hình dưới đây.



Hình 3: Các phép toán tập hợp.

Ví dụ, ta muốn lọc các chuyến bay bị chậm trễ thời gian cất cánh trong tháng 3, lớn hơn hoặc bằng 30 phút. Lúc này ta cần áp dụng điều kiện trên hai features là `dep_delay` và `month`:

```
flights |> filter(dep_delay >= 30 & month == 3)
```

```
## # A tibble: 4,402 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     3     1        4           2159       125      318             56
## 2  2013     3     1       50           2358        52      526            438
## 3  2013     3     1      117           2245       152      223           2354
## 4  2013     3     1      653            600        53      756            724
## 5  2013     3     1      716            630        46     1010           1019
## # i 4,397 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Như vậy, ta có 4402 chuyến bay cất cánh trễ ít nhất 30 phút, trong tháng 1.

Một ví dụ khác, nếu ta muốn lọc ra các chuyến bay trong tháng 6 hoặc tháng 7, có thể dùng

```
flights |> filter(month == 6 | month == 7)
```

```
## # A tibble: 57,668 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
```

```
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     6     1        2           2359         3      341            350
## 2  2013     6     1      451            500        -9      624            640
## 3  2013     6     1      506            515        -9      715            800
## 4  2013     6     1      534            545       -11      800            829
## 5  2013     6     1      538            545        -7      925            922
## # i 57,663 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Ngoài ra, ta có thể dùng toán tử `%in%` để thay thế cho `month == 6 | month == 7`.

```
flights |> filter(month %in% c(6, 7))
```

```
## # A tibble: 57,668 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     6     1        2           2359         3      341            350
## 2  2013     6     1      451            500        -9      624            640
## 3  2013     6     1      506            515        -9      715            800
## 4  2013     6     1      534            545       -11      800            829
## 5  2013     6     1      538            545        -7      925            922
## # i 57,663 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

tức là lọc ra các quan sát của các tháng (`month`) có trong một tập xác định.

Trong một số trường hợp, ta có thể cần lọc ra các dòng dữ liệu duy nhất (không bị lặp lại) của bảng dữ liệu. Để làm điều này, ta có thể sử dụng hàm `distinct()`. Nếu ta sử dụng cú pháp:

```
data_name |> distinct()
```

tức là ta đang áp dụng hàm `distinct()` cho tất cả cột của dữ liệu, ngược lại, nếu ta thêm vào tên của 1 hoặc nhiều cột của bảng dữ liệu, thì ta đang tìm các giá trị duy nhất của một biến hoặc kết hợp nhiều biến. Ví dụ,

```
flights |> distinct(month)
```

```
## # A tibble: 12 x 1
##    month
##    <int>
## 1      1
## 2     10
## 3     11
## 4     12
## 5      2
## # i 7 more rows
```

ta thu được các tháng được thu thập dữ liệu trong năm. Hoặc một ví dụ khác,

```
flights |> distinct(origin, dest)
```

```
## # A tibble: 224 x 2
##    origin dest
##    <chr>  <chr>
## 1 EWR    IAH
## 2 LGA    IAH
```

```
## 3 JFK    MIA
## 4 JFK    BQN
## 5 LGA    ATL
## # i 219 more rows
```

giúp ta lọc ra các cặp địa điểm xuất phát và điểm đến, duy nhất. Ta có thể thêm đối số `.keep_all = TRUE`, thì ta thu được một bảng dữ liệu với các cột được lọc theo các dòng duy nhất tương ứng với biến được sử dụng:

```
flights |> distinct(month, .keep_all = TRUE)
```

```
## # A tibble: 12 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013    10     1      447            500       -13      614            648
## 3  2013    11     1        5           2359         6      352            345
## 4  2013    12     1       13           2359        14      446            445
## 5  2013     2     1      456            500        -4      652            648
## # i 7 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Để sắp xếp thứ tự các giá trị trong một cột theo chiều tăng dần, ta có thể dùng hàm `arrange()`. Ví dụ:

```
flights |> arrange(dep_time)
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1    13        1           2249        72      108           2357
## 2  2013     1    31        1           2100       181      124           2225
## 3  2013    11    13        1           2359         2      442            440
## 4  2013    12    16        1           2359         2      447            437
## 5  2013    12    20        1           2359         2      430            440
## # i 336,771 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

ta thu được bảng dữ liệu mới, trong đó, các dòng được sắp xếp tương ứng theo giá trị tăng dần của thời gian cất cánh `dep_time`, giá trị thấp nhất là 1 (00:01). Ngược lại, nếu ta muốn sắp xếp giảm dần, ta cần thêm hàm `desc()` vào trong hàm `arrange()`. Ví dụ:

```
flights |> arrange(desc(dep_time))
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013    10    30     2400           2359         1      327            337
## 2  2013    11    27     2400           2359         1      515            445
## 3  2013    12     5     2400           2359         1      427            440
## 4  2013    12     9     2400           2359         1      432            440
## 5  2013    12     9     2400           2250        70       59           2356
## # i 336,771 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
```

```
## #   minute <dbl>, time_hour <dttm>
```

Nếu ta cung cấp nhiều tên cột, mỗi cột bổ sung sẽ được sử dụng để phá vỡ mối liên kết trong các giá trị của các cột trước đó. Ví dụ: đoạn code sau sắp xếp theo thời gian khởi hành `dep_time`, được trải rộng trên bốn cột. Đầu tiên chúng ta nhận được những năm sớm nhất, sau đó trong vòng một năm là những tháng sớm nhất, v.v.

```
flights |> arrange(year, month, day, dep_time)
```

```
## # A tibble: 336,776 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## # i 336,771 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Tức là ta thu được bảng dữ liệu sắp xếp các chuyến bay đầu tiên trong ngày/tháng/năm.

2.3.2 Các thao tác với cột của dữ liệu

Để tạo một cột biến mới, ta sử dụng hàm `mutate()`:

```
data_name_new <- data_name |> mutate(ten_bien = gia_tri)
```

Ví dụ

```
flights_2 <- flights |> mutate(gain = dep_delay - arr_delay)
glimpse(flights_2)
```

```
## Rows: 336,776
## Columns: 20
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1, 0, ~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849, 853,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851, 856,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -14, 31~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA", ~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 49, 71~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39463", ~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "JFK~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "MCO~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 158, 3~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, 1028,~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, ~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0, 0, ~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 05:00:~
## $ gain           <dbl> -9, -16, -31, 17, 19, -16, -24, 11, 5, -10, 0, 1, -9, 12, -3~
```

Ta có thể tạo nhiều hơn 1 cột cùng một lúc:

```
flights_2 <- flights |> mutate(gain = dep_delay - arr_delay,
                               speed = distance / air_time * 60)
glimpse(flights_2)
```

```
## Rows: 336,776
## Columns: 21
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1, 0, ~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849, 853,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851, 856,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -14, 31~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "AA", ~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 49, 71~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39463", ~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA", "JFK~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD", "MCO~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 158, 3~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, 1028,~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6, 6, ~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0, 0, ~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 05:00:~
## $ gain           <dbl> -9, -16, -31, 17, 19, -16, -24, 11, 5, -10, 0, 1, -9, 12, -3~
## $ speed          <dbl> 370.0441, 374.2731, 408.3750, 516.7213, 394.1379, 287.6000, ~
```

Thông thường, các biến mới tạo, sẽ được thêm vào phía sau của bảng dữ liệu, do đó, để thuận tiện cho việc kiểm tra, ta có thể thêm đối số `.before = 1` để chèn các cột mới vào phía trước của bảng dữ liệu

```
flights_3 <- flights |> mutate(gain = dep_delay - arr_delay,
                               speed = distance / air_time * 60,
                               .before = 1)
flights_3
```

```
## # A tibble: 336,776 x 21
##    gain   speed  year month   day dep_time sched_dep_time dep_delay arr_time
##   <dbl>   <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1    -9 370.044  2013     1     1      517            515         2      830
## 2   -16 374.273  2013     1     1      533            529         4      850
## 3   -31 408.375  2013     1     1      542            540         2      923
## 4    17 516.721  2013     1     1      544            545        -1     1004
## 5    19 394.138  2013     1     1      554            600        -6      812
## # i 336,771 more rows
## # i 12 more variables: sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Trong một số trường hợp phân tích, ta chỉ cần sử dụng một số features (biến), để tạo ra một dữ liệu con chứa các biến được lựa chọn, ta có thể sử dụng `select()`:

- lựa chọn theo tên của biến

```
flights_4 <- flight |> select(year, month, day)
```

- lựa chọn tất cả các cột nằm giữa cột `year` và `day`:

```
flights_4 <- flight |> select(year:day)
```

- lựa chọn các cột không nằm giữa cột `year` và `day`:

```
flights_4 <- flight |> select(!year:day)
```

- lựa chọn tất cả các cột là dạng chữ:

```
flights_4 <- flights |> select(where(is.character))
```

Trong một số trường hợp, ta có thể đổi tên của cột bằng hàm `rename()`:

```
flights |> rename(depart_time = dep_time)
```

```
## # A tibble: 336,776 x 19
##    year month   day depart_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>       <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1         517            515         2      830            819
## 2  2013     1     1         533            529         4      850            830
## 3  2013     1     1         542            540         2      923            850
## 4  2013     1     1         544            545        -1     1004           1022
## 5  2013     1     1         554            600        -6      812            837
## # i 336,771 more rows
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Để di chuyển các cột, ta có thể dùng hàm `relocate()`.

```
flights |> relocate(air_time)
```

```
## # A tibble: 336,776 x 19
##   air_time  year month   day dep_time sched_dep_time dep_delay arr_time
##      <dbl> <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1      227  2013     1     1      517            515         2      830
## 2      227  2013     1     1      533            529         4      850
## 3      160  2013     1     1      542            540         2      923
## 4      183  2013     1     1      544            545        -1     1004
## 5      116  2013     1     1      554            600        -6      812
## # i 336,771 more rows
## # i 11 more variables: sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Mặc định, hàm `relocate()` sẽ di chuyển cột mục tiêu lên vị trí đầu tiên, tuy nhiên, ta có thể ấn định vị trí này

- trước một cột

```
flights |> relocate(air_time, .before = dep_time)
```

- sau một cột

```
flights |> relocate(air_time, .after = day)
```

Ta cũng có thể di chuyển một cụm gồm nhiều cột một lúc

```
flights |> relocate(air_time, origin, dest, distance)
```

```
## # A tibble: 336,776 x 19
##   air_time origin dest  distance  year month   day dep_time sched_dep_time dep_delay
##      <dbl> <chr>  <chr>    <dbl> <int> <int> <int>    <int>          <int>     <dbl>
## 1      227 EWR    IAH       1400  2013     1     1      517            515         2
## 2      227 LGA    IAH       1416  2013     1     1      533            529         4
## 3      160 JFK    MIA       1089  2013     1     1      542            540         2
## 4      183 JFK    BQN       1576  2013     1     1      544            545        -1
## 5      116 LGA    ATL        762  2013     1     1      554            600        -6
## # i 336,771 more rows
## # i 9 more variables: arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

3 Bài tập

Bài tập 1: Trong một quy trình duy nhất cho từng điều kiện, hãy tìm tất cả các chuyến bay đáp ứng điều kiện:

- Đến nơi trễ từ hai giờ trở lên
- Bay tới Houston (IAH hoặc HOU)
- Được điều hành bởi United, American hoặc Delta
- Khởi hành vào mùa hè (tháng 7, tháng 8, tháng 9)
- Đến muộn hơn hai tiếng nhưng không cất cánh muộn
- Bị trì hoãn ít nhất một giờ nhưng lại kéo dài hơn 30 phút trên chuyến bay

Bài tập 2: Sắp xếp `flights` để tìm chuyến bay có thời gian khởi hành trễ nhất. Tìm các chuyến bay khởi hành sớm nhất vào buổi sáng.

Bài tập 3: Sắp xếp `flights` để tìm chuyến bay có vận tốc nhanh nhất. (Gợi ý: Hãy thử đưa phép tính toán vào bên trong hàm của bạn.)

Bài tập 4: Các chuyến bay hàng ngày trong năm 2013, đúng hay không?

Bài tập 5: Chuyến bay nào đi được quãng đường xa nhất? Chuyến nào đã đi được quãng đường ngắn nhất?

Bài tập 6: So sánh các biến `dep_time`, `sched_dep_time` và `dep_delay`, liệu chúng có mối liên hệ nào với nhau.

Bài tập 7: Tìm hiểu các hàm `starts_with`, `ends_with`, `contains()`. Áp dụng chúng vào trong nhiệm vụ lựa chọn các cột `dep_time`, `dep_delay`, `arr_time` và `arr_delay`.

Bài tập 8: Hàm `any_of()` làm gì? Tại sao nó có thể hữu ích khi kết hợp với vectơ này?

```
variables <- c("year", "month", "day", "dep_delay", "arr_delay")
```

Bài tập 9: Tại sao đoạn chương trình sau không hoạt động? Lỗi được báo có ý nghĩa gì?

```
flights |> select(tailnum) |>
  arrange(arr_delay)
```

```
## Error in `arrange()`:
## i In argument: `..1 = arr_delay`.
## Caused by error:
## ! object 'arr_delay' not found
```

Bài tập 10: Đổi tên cột `air_time` thành `air_time_min` để chỉ rõ đơn vị đo, đồng thời, di chuyển cột này về vị trí bắt đầu của bảng dữ liệu.