Bài giảng R, số 2
-Khám phá phân tích dữ liệu với R, p2-

TS.Tô Đức Khánh

04/03/2024

1 Bảng dữ liệu tổng hợp cho biến định lượng

1.1 Thống kê tổng hợp cho một biến định lượng

Đầu tiên, ta sẽ tính giá trị trung bình và độ lệch chuẩn cho một biến định lượng trong dữ liệu.

Xét dữ liệu liệu `flights` trong thư viện `nycflights13`, bao hàm thông tin các chuyến bay cất cánh từ thành phố New York trong năm 2013.

```
library(nycflights13)
data(flights)
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, 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 tạo một bảng báo cáo bao gồm trung bình và độ lệch chuẩn của `bwt`, ta sử dụng hàm `summarise()` kết hợp các hàm cơ bản của `R`, ví dụ: `mean()` và `sd()`. Ngoài ra, ta cũng có thể gọi các hàm tính trung vị (*median*), giá trị nhỏ nhất, giá trị lớn nhất, khoảng tứ phân vị trong *interquartile range*:

```
flights |> summarise(tb = mean(dep_delay), dlc = sd(dep_delay))
```

```
## # A tibble: 1 x 2
##      tb   dlc
```

```
##   <dbl> <dbl>
## 1    NA    NA
```

Kết quả thu được là `NA`, bởi lẽ có dữ liệu khuyết trong `dep_delay`. Để loại đi giá trị khuyết trong quá trình tính, ta thêm đối số `na.rm = TRUE` vào trong các hàm `mean()` và `sd()`:

```
flights |> summarise(tb = mean(dep_delay, na.rm = TRUE),
                     dlc = sd(dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 2
##        tb     dlc
##     <dbl>   <dbl>
## 1 12.6391 40.2101
```

Các hàm cơ bản trong `R`:

- `median()`: tính trung vị của một vector dữ liệu;
- `IQR()`: tính khoảng tứ phân vị;
- `min()`: xác định giá trị nhỏ nhất;
- `max()`: xác định giá trị lớn nhất;
- `mad()`: tính chỉ số meadian absolute deviation (MAD);
- `weighted.mean()`: tính trung bình có trọng số.

Ngoài ra, thư viện `matrixStats` còn cung cấp một số hàm hữu dụng khác

- `weightedMedian()`: tính trung vị có trọng số;
- `weightedMad()`: tính chỉ số MAD có trọng số;
- `weightedSd()`: tính độ lệch chuẩn có trọng số;
- `weightedVar()`: tính phương sai có trọng số.

Bài tập 1: Xét cột `dep_delay`, `arr_delay`, `air_time` và `distance`. Hãy tạo bảng tóm tắt cho một biến, có chứa giá trị trung vị, giá trị lớn nhất, giá trị nhỏ nhất, khoảng tứ phân vị.

Bài tập 2: Dữ liệu `state.csv` cung cấp tỷ lệ tội phạm giết người và dân số của 53 bang của Mỹ. Hãy tạo một bảng số liệu tổng hợp cho `population` và cho `murder_rate`.

1.2 Thống kê tổng hợp cho nhiều biến định lượng

Trong phần này, ta sử dụng dữ liệu `birthwt.txt` về cân nặng của trẻ sơ sinh ở Mỹ. Biến `bwt` biểu thị cân nặng của trẻ khi sinh (đơn vị: grams).

```
data_birth <- read_table(file = "datasets/birthwt.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   low = col_double(),
##   age = col_double(),
##   lwt = col_double(),
##   race = col_double(),
##   smoke = col_double(),
##   ptl = col_double(),
##   ht = col_double(),
##   ui = col_double(),
##   ftv = col_double(),
```

```
##   bwt = col_double()
## )
```

```
data_birth <- data_birth |>
  mutate(race = factor(race, labels = c("white", "black", "other")),
         smoke = factor(smoke, labels = c("no", "yes")))
glimpse(data_birth)
```

```
## Rows: 189
## Columns: 10
## $ low   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ age   <dbl> 19, 33, 20, 21, 18, 21, 22, 17, 29, 26, 19, 19, 22, 30, 18, 18, 15, 2~
## $ lwt   <dbl> 182, 155, 105, 108, 107, 124, 118, 103, 123, 113, 95, 150, 95, 107, 1~
## $ race  <fct> black, other, white, white, white, other, white, other, white, white,~
## $ smoke <fct> no, no, yes, yes, yes, no, no, no, yes, yes, no, no, no, no, yes, yes~
## $ ptl   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ ht    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ ui    <dbl> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, ~
## $ ftv   <dbl> 0, 3, 1, 2, 0, 0, 1, 1, 1, 0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 1, 2, 3, 1, ~
## $ bwt   <dbl> 2523, 2551, 2557, 2594, 2600, 2622, 2637, 2637, 2663, 2665, 2722, 273~
```

Trong các báo cáo thống kê, ta thường phải trình bày bảng thống kê tổng hợp cho nhiều biến định lượng cùng một lúc. Để làm điều này một cách nhanh chóng và có kết quả trình bày ưu nhìn, ta sẽ thực hiện theo hai bước như sau:

Bước 1. Sử dụng hàm `summarise()` kết hợp với `across()` để tính các ước lượng thống kê, kết quả sau đó được lưu lại trong một biến,

```
df_sum <- data_birth |>
  summarise(across(c("bwt", "lwt", "age"),
                   list(gtnn = min, gtln = max, tv = median,
                        tb = mean, dlc = sd)))
```

trong hàm `across()`, ta cần 2 đối số:

- tên của các biến cần tính;
- danh sách các hàm cần dùng để tính các ước lượng thống kê mong muốn.

Ở đây, ta muốn tính giá trị nhỏ nhất (gtnn), giá trị lớn nhất (gtln), trung vị (tv), trung bình (tb) và độ lệch chuẩn (dlc). Bảng kết quả thô sẽ là

```
df_sum
```

```
## # A tibble: 1 x 15
##   bwt_gtnn bwt_gtln bwt_tv  bwt_tb bwt_dlc lwt_gtnn lwt_gtln lwt_tv  lwt_tb lwt_dlc
##      <dbl>    <dbl>  <dbl>   <dbl>   <dbl>    <dbl>    <dbl>  <dbl>   <dbl>   <dbl>
## 1      709     4990   2977 2944.66 729.022       80      250    121 129.815 30.5794
## # i 5 more variables: age_gtnn <dbl>, age_gtln <dbl>, age_tv <dbl>, age_tb <dbl>,
## #   age_dlc <dbl>
```

Bước 2. Để tạo ra một bảng kết quả dễ nhìn hơn, ta dùng đoạn lệnh sau:

```
df_stats_tidy <- df_sum |> gather(ten, gt) |>
  separate(ten, into = c("bien", "tk"), sep = "_") |>
  spread(tk, gt) |>
  select(bien, gtnn, gtln, tv, tb, dlc)
print.data.frame(df_stats_tidy, digits = 4)
```

```
##   bien gtnn gtln   tv      tb     dlc
```

```
## 1  age   14   45   23   23.24   5.299
## 2  bwt  709 4990 2977 2944.66 729.022
## 3  lwt   80  250  121  129.81  30.579
```

Ý tưởng của đoạn lệnh trên như sau:

- Sử dụng `gather()` để tạo ra một bảng dữ liệu với hai cột có tên là `ten` (tên) và `gt` (giá trị).

```
df_sum |> gather(ten, gt)
```

```
## # A tibble: 15 x 2
##   ten            gt
##   <chr>       <dbl>
## 1 bwt_gtnn  709
## 2 bwt_gtln 4990
## 3 bwt_tv   2977
## 4 bwt_tb   2944.66
## 5 bwt_dlc   729.022
## # i 10 more rows
```

- Thành phần của cột `ten` bao gồm hai thành phần, được ngăn cách nhau bởi dấu `_`, thành phần phía trước là tên biến được quan tâm, thành phần phía sau là tên của ước lượng thống kê cần tính (gtnn, gtln, tb, tv, dlc). Với tính chất này, ta có thể tách `ten` thành hai cột mới, đặt là `"bien"` và `"tk"`, lần lượt biểu thị tên của biến và tên của ước lượng thống kê. Để làm điều này, ta dùng hàm `separate()`:

```
df_sum |> gather(ten, gt) |>
  separate(ten, into = c("bien", "tk"), sep = "_")
```

```
## # A tibble: 15 x 3
##   bien  tk          gt
##   <chr> <chr>    <dbl>
## 1 bwt   gtnn   709
## 2 bwt   gtln  4990
## 3 bwt   tv    2977
## 4 bwt   tb    2944.66
## 5 bwt   dlc    729.022
## # i 10 more rows
```

- Từ bảng mới này, ta dùng hàm `spread()` cho hai biến `tk` và `gt`, để tạo ra bảng hai chiều (tương tự như ta đã làm cho bảng tần số chéo):

```
df_sum |> gather(ten, gt) |>
  separate(ten, into = c("bien", "tk"), sep = "_") |>
  spread(tk, gt)
```

```
## # A tibble: 3 x 6
##   bien        dlc  gtln  gtnn        tb    tv
##   <chr>     <dbl> <dbl> <dbl>     <dbl> <dbl>
## 1 age     5.29868    45    14   23.2381    23
## 2 bwt   729.022    4990   709 2944.66    2977
## 3 lwt    30.5794    250    80  129.815    121
```

- Tuy nhiên, thứ tự các cột hơi lộn xộn, ta có thể sắp xếp lại theo thứ tự mà mình mong muốn, bằng cách sử dụng hàm `select()`:

```
df_sum |> gather(ten, gt) |>
  separate(ten, into = c("bien", "tk"), sep = "_") |>
  spread(tk, gt) |>
  select(bien, gtnn, gtln, tv, tb, dlc)
```

```
## # A tibble: 3 x 6
##   bien   gtnn  gtln    tv        tb       dlc
##   <chr> <dbl> <dbl> <dbl>     <dbl>     <dbl>
## 1 age      14    45    23   23.2381   5.29868
## 2 bwt     709  4990  2977 2944.66   729.022
## 3 lwt      80   250   121  129.815   30.5794
```

- Kết quả sau đó được lưu trữ vào một biến. Để in kết quả, ta dùng hàm `print.data.frame()` với đối số `digits = 4` để điều chỉnh số chữ số sau dấu thập phân được hiển thị.

Bài tập 3: Tạo bảng kết quả tổng hợp cho các biến `dep_delay`, `arr_delay`, `air_time`, theo các tháng khác nhau.

Bài tập 4: Tệp dữ liệu `Pima.csv` ghi nhận lại các chỉ số sinh học khác nhau của 532 nữ thổ dân da đỏ Pima, độ tuổi từ 21 trở lên, sống tại Phoenix, Arizona, Mỹ. Những người này cũng được kiểm tra bệnh đái tháo đường theo tiêu chuẩn của tổ chức y tế thế giới (WHO). Các biến được ghi nhận như sau:

- `npreg`: số lần mang thai;
- `glu`: nồng độ plasma glucose trong xét nghiệm dung nạp glucose đường uống;
- `bp`: huyết áp tâm trương (mm Hg);
- `skin`: độ dày nếp gấp da cơ tam đầu - bắp thịt to ở đằng sau cánh tay trên (mm);
- `bmi`: chỉ số mật độ cơ thể (cân nặng (kg) chia cho bình phương chiều cao (m));
- `ped`: khả năng bị bệnh tiểu đường theo phả hệ;
- `age`: độ tuổi (năm);
- `type`: trạng thái bị bệnh tiểu đường (yes/no);

Hãy thực hiện các thao tác sau:

1. Nhập dữ liệu này vào trong không gian làm việc của `R`, sau đó, kiểm tra xem liệu các tên biến, loại biến đã đúng theo quy chuẩn chưa? Nếu chưa hãy hiệu chỉnh lại cho đúng.
2. Lập bảng thống kê tổng hợp cho 1 biến định lượng bất kỳ.
3. Lập bảng thống kê tổng hợp cho 1 vài biến định lượng bất kỳ.

1.3 Thống kê tổng hợp cho biến định lượng phân theo nhóm

Trong một số trường hợp, các đối tượng trong nghiên cứu có thể được phân thành các nhóm theo một tính chất hay biến định tính nào đó, ví dụ:

- giới tính;
- nhóm tuổi;
- tình trạng bệnh;
- trạng thái hút thuốc lá.

Khi đó, các ước lượng thống kê tổng hợp của biến định lượng cũng có thể sẽ khác nhau theo từng nhóm đối tượng. Trong phần này, ta sẽ học cách tạo ra một bảng thống kê tổng hợp cho 1 hoặc nhiều biến định lượng, khi đối tượng được phân theo nhóm.

Đối với trường hợp 1 biến định lượng, ta vẫn sẽ sử dụng hàm `summarise()`, tuy nhiên, trước đó, ta sẽ dùng hàm `group_by()` để phân đối tượng thành các nhóm dựa trên thông tin của một hoặc hai biến phân loại. Ví dụ, ta muốn tạo bảng thống kê tổng hợp, bao gồm thông tin của số lượng đối tượng, trung bình và độ lệch chuẩn, cho `bwt` cho các nhóm chủng tộc khác nhau được ghi nhận trong dữ liệu `birthwt.txt`, cấu trúc đoạn lệnh sẽ như sau:

```r
data_birth |> group_by(race) |>
  summarise(n = n(), tb = mean(bwt), dlc = sd(bwt))
```

```
## # A tibble: 3 x 4
##   race      n      tb     dlc
##   <fct> <int>   <dbl>   <dbl>
## 1 white    96 3103.74 727.724
## 2 black    26 2719.69 638.684
## 3 other    67 2804.01 721.301
```

trong đó, `group_by(race)` sẽ phân chia đối tượng thành ba nhóm: `white`, `black` và `other`. Hàm `n()` dùng để đếm số lượng đối tượng.

Bài tập 5: thực hiện các thao tác sau

- tạo bảng thống kê tổng hợp cho biến `bwt` theo các nhóm của trạng thái hút thuốc của mẹ;
- tạo bảng thống kê tổng hợp cho biến `bwt` theo các nhóm của chủng tộc và trạng thái hút thuốc của mẹ;
- lập lại các bảng trên, nhưng thêm một cột tính độ lệch chuẩn của trung bình mẫu.

Bài tập 6: Xét dữ liệu `flights`. Hãy tạo các bảng tổng hợp phù hợp để trả lời các câu hỏi sau:

- Số phút cất cánh trễ của chuyến bay là giống nhau cho các sân bay đi?
- Liệu sự trễ chuyến bay có bị ảnh hưởng bởi các tháng trong năm? Tìm các tháng có chuyến bay trễ nhất hoặc ít trễ nhất.
- Các hãng hàng không có phải nguyên nhân khiến các chuyến bay cất cánh trễ? Tìm các hãng hàng không tệ nhất. Liệu sự cất cánh trễ của các hãng hàng không này, có khác biệt giữa các sân bay xuất phát?

Trong trường hợp, ta muốn làm bảng thống kê tổng hợp cho nhiều hơn một biến định lượng, ta cũng sẽ làm tương tự như trường hợp không phân nhóm. Lưu ý, khi tạo bảng kết quả, hàm `gather()` sẽ cần chỉnh sửa để không sử dụng cột chứa thông tin của nhóm được chia. Cụ thể, trong trường hợp này, cột `race` sẽ không được dùng, khi đó cấu trúc `gather()` sẽ là `gather(ten, gt, -race)`, dấu `"-"` có nghĩa là "loại bỏ".

```r
df_sum_race <- data_birth |> group_by(race) |>
  summarise(across(c("bwt", "lwt", "age"),
                   list(gtnn = min, gtln = max, tv = median,
                        tb = mean, dlc = sd)))
df_stats_tidy_race <- df_sum_race |> gather(ten, gt, -race) |>
  separate(ten, into = c("bien", "tk"), sep = "_") |>
  spread(tk, gt) |>
  select(race, bien, gtnn, gtln, tv, tb, dlc)
print.data.frame(df_stats_tidy_race, digits = 4)
```

```
##    race bien gtnn gtln     tv      tb     dlc
## 1 white  age   14   45   23.5   24.29   5.655
## 2 white  bwt 1021 4990 3076.0 3103.74 727.724
## 3 white  lwt   90  235  129.5  132.05  29.094
## 4 black  age   15   35   20.5   21.54   5.109
## 5 black  bwt 1135 3860 2849.0 2719.69 638.684
## 6 black  lwt   98  241  129.0  146.81  39.639
## 7 other  age   14   33   22.0   22.39   4.536
## 8 other  bwt  709 4054 2835.0 2804.01 721.301
## 9 other  lwt   80  250  119.0  120.01  25.130
```

2 Biểu diễn cho biến định lượng

Trong phần này, ta sẽ học cách biểu diễn dữ liệu bằng các biểu đồ với thư viện `ggplot2` (xem chi tiết về vẽ hình với `ggplot2` tại chương 2, của sách bản online). Về cơ bản, cấu trúc của một đoạn lệnh vẽ biểu đồ với `ggplot2` như sau:

```
ggplot(data = ten_du_lieu,
       mapping = aes(x = ten_bien_o_truc_x, y = ten_bien_truc_y,
                     cac_doi_so_khac)) +
  \\ các hàm vẽ đối tượng: histogram, boxplot, bar_plot +
  \\ các hàm hiệu chỉnh hình
```

Trong một số trường hợp, ta có thể chỉ sử dụng `x` hoặc `y`. `cac_doi_so_khac` có thể là:

- `fill`: tô màu bên trong các khối của hình;
- `color`: tô màu đường viền;
- `shape`: hiệu chỉnh hình dạng của điểm.

Tùy vào từng tình huống mà ta sẽ sử dụng các thông số này. Dấu `"+"` trong đoạn lệnh trên là để ghép nối các câu lệnh vẽ với nhau tạo thành một khối vẽ `ggplot`.

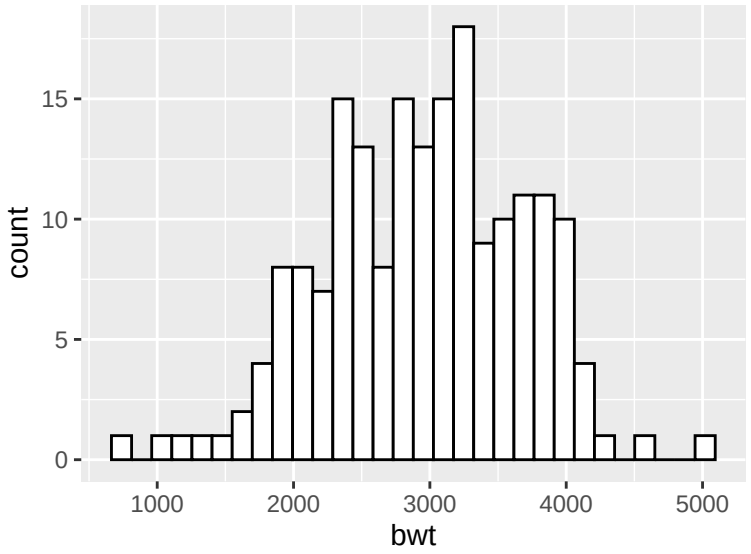
2.1 Biểu đồ tần số và biểu đồ hộp cho 1 biến định lượng

Biểu đồ tần số được tạo bởi hàm `geom_histogram()`, trong kh đó, hàm `geom_boxplot()` sẽ tạo biểu đồ hộp.

Để vẽ biểu đồ tần số cho cân nặng của trẻ sơ sinh, `bwt`, trong bộ dữ liệu `data_birth`, một cách đơn giản, ta thực hiện như sau:

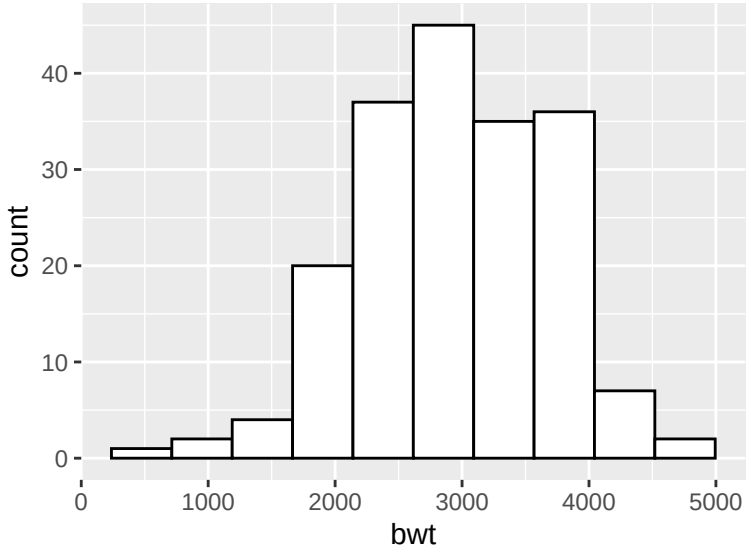
```
ggplot(data = data_birth, aes(x = bwt)) +
  geom_histogram(fill = "white", color = "black")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Theo mặc định, hàm `geom_histogram()` sẽ tự động tính số khoảng tần số (thông qua hàm `stat_bin()`), kết quả thường bị quá nhiều khoảng, dẫn tới biểu đồ không được dễ nhìn (do quá chi tiết). Để khắc phục điều này, ta có thể điều chỉnh số khoảng tần số bằng tham số `bins`, chẳng hạn, `bins = 10`:

```r
ggplot(data = data_birth, aes(x = bwt)) +
  geom_histogram(fill = "white", color = "black", bins = 10)
```

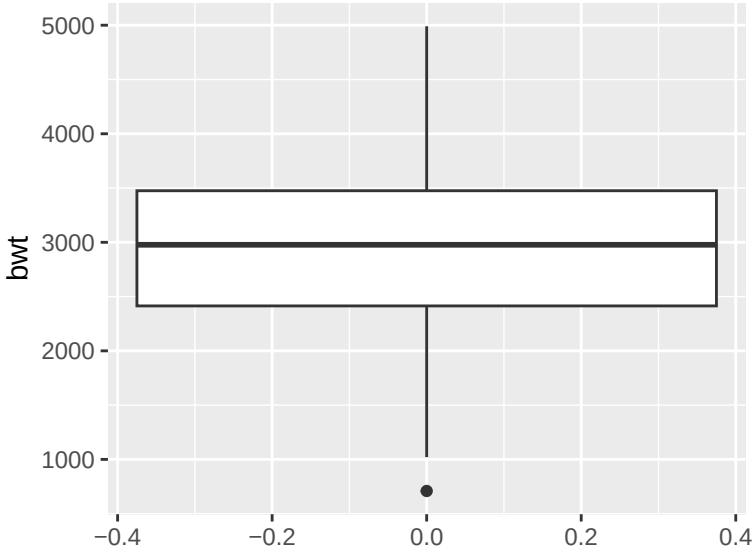


Ngoài ra, ta cũng có thể dùng:

- `binwidth` để hiểu chỉnh độ rộng của từng khoảng tần số; hoặc
- `breaks` để định nghĩa các khoảng tần số.

Biểu đồ boxplot được tạo như sau:

```r
ggplot(data = data_birth, aes(y = bwt)) +
  geom_boxplot()
```



Bài tập 7: thực hiện các thao tác sau:

- vẽ lại biểu đồ tần số và biểu đồ boxplot cho biến `bwt`, thay đổi tên các trục số (sử dụng hàm `xlab()`, `ylab()` hoặc `labs()`), hiệu chỉnh màu (sử dụng đối số `color = "ten_mau"`) và hình nền (tìm hiểu các hàm `theme_bw()`, `theme_classic()` và một số hàm tương tự);

- sử dụng hàm `geom_vline()` hoặc hàm `geom_hline()` để tạo một đường thẳng biểu thị giá trị trung bình của `bwt`, chỉnh màu đường thẳng và kiểu đường thẳng, bằng hai đối số `linetype` và `color`;
- vẽ biểu đồ tần số và biểu đồ boxplot cho biến `age` trong dữ liệu `data_birth`, thay đổi tiêu đề tên trục số và màu sắc của biểu đồ;
- sử dụng hàm `geom_vline()` hoặc hàm `geom_hline()` để tạo một đường thẳng biểu thị giá trị trung bình của `age`, chỉnh màu đường thẳng và kiểu đường thẳng;
- nếu thêm `coord_flip()` thì ta thu được gì?

2.2 Ước lượng hàm mật độ xác suất

Để miêu tả ước lượng hàm mật độ xác suất của một biến định lượng, ta sử dụng hàm `geom_density()`:

```
ggplot(ten_du_lieu, aes(x = ten_bien_o_truc_x)) +
  geom_density(color = "blue", bw = ..., kernel = ...) +
  \\ các hàm hiệu chỉnh hình
```

Trong đó, `cac_doi_so_khac` có thể là:

- `ten_bien_o_truc_x`: tương ứng là biến cần tính mật độ;
- `color`: tô màu đường viền;
- `bw`: thông số hiệu chỉnh bandwidth;
- `kernel`: tên hàm kernel sử dụng trong ước lượng.

Đối với bandwidth, ta có các lựa chọn sau:

- `bw = "nrd0"`: lựa chọn bandwidth theo phương pháp rule-of-thumb với $C = 0.9$;
- `bw = "nrd"`: lựa chọn bandwidth theo phương pháp rule-of-thumb với $C = 1.06$;
- `bw = "ucv"`: lựa chọn bandwidth theo phương pháp unbiased cross-validation;
- `bw = "bcv"`: lựa chọn bandwidth theo phương pháp biased cross-validation;
- `bw = "SJ"`: lựa chọn bandwidth theo phương pháp Sheather & Jones;
- `bw = a`: ấn định một giá trị cụ thể cho bandwidth.

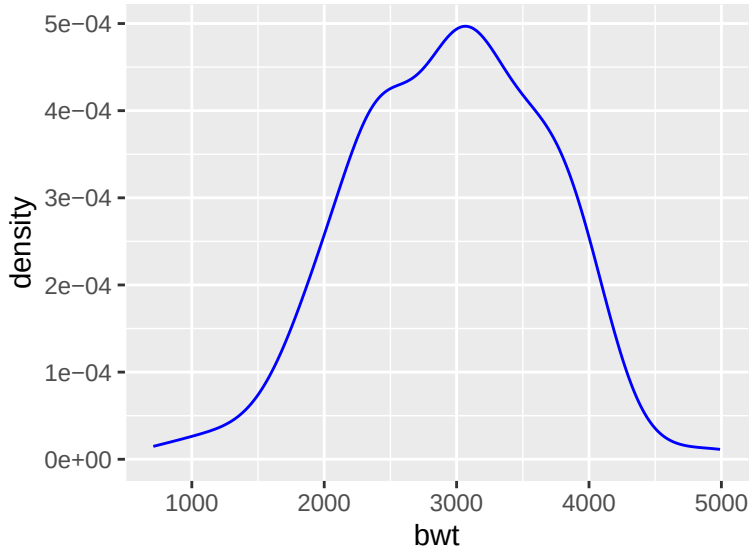
Đối với tên hàm kernel[1], ta có các lựa chọn sau:

- `kernel = "gaussian"`: lựa chọn Gaussian kernel;
- `kernel = "epanechnikov"`: lựa chọn Epanechnikov kernel;
- `kernel = "rectangular"`: lựa chọn Rectangular kernel;
- `kernel = "triangular"`: lựa chọn Triangular kernel;
- `kernel = "biweight"`: lựa chọn Biweight kernel;
- `kernel = "cosine"`: lựa chọn cosine kernel;
- `kernel = "optsine"`: lựa chọn optsine kernel;

Ví dụ, ta ước lượng hàm mật độ xác suất cho cân nặng của trẻ sơ sinh:

```
ggplot(data_birth, aes(x = bwt)) +
  geom_density(color = "blue", bw = "nrd0", kernel = "gaussian")
```

[1] chi tiết các hàm kernel, xem tại: https://en.wikipedia.org/wiki/Kernel_(statistics)



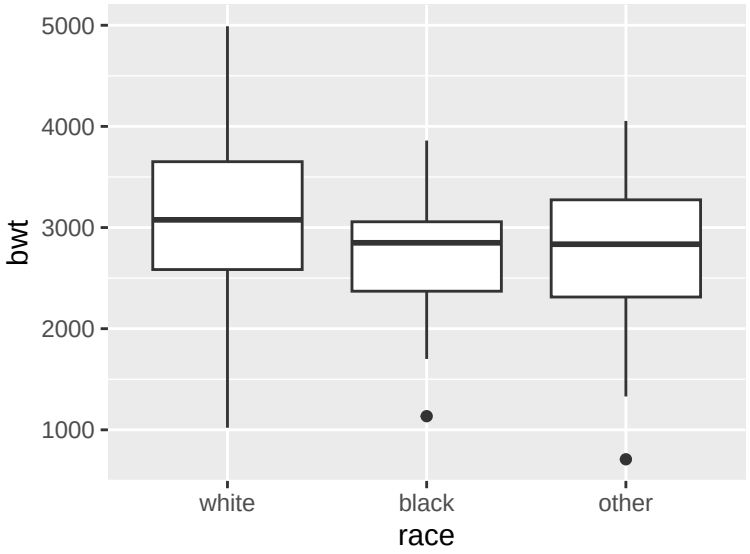
Bài tập 8:

- Tìm hiểu hàm `geom_rug()`. Nếu kết hợp với đoạn vẽ ước lượng hàm mật độ, kết quả thu được như thế nào?
- Kết hợp histogram và biểu đồ hàm mật độ. Chú ý, cần sử dụng `aes(y = ..density..)` trong hàm `geom_histogram()` để biểu diễn tần số tương đối thay cho tần số tuyệt đối.
- Biểu diễn ước lượng mật độ xác suất của cân nặng trẻ sơ sinh tương ứng với hai trạng thái hút thuốc của người mẹ. (Hai biểu đồ riêng biệt).
- Sử dụng hàm `geom_vline()` để vẽ thêm đường thẳng mô tả trung bình của cân nặng của trẻ sơ sinh. Điều chỉnh kiểu đường và màu sắc.
- Thử với các cách chọn bandwidth và kernel khác nhau.

2.3 Biểu diễn mối liên hệ của 1 biến định lượng và biến định tính

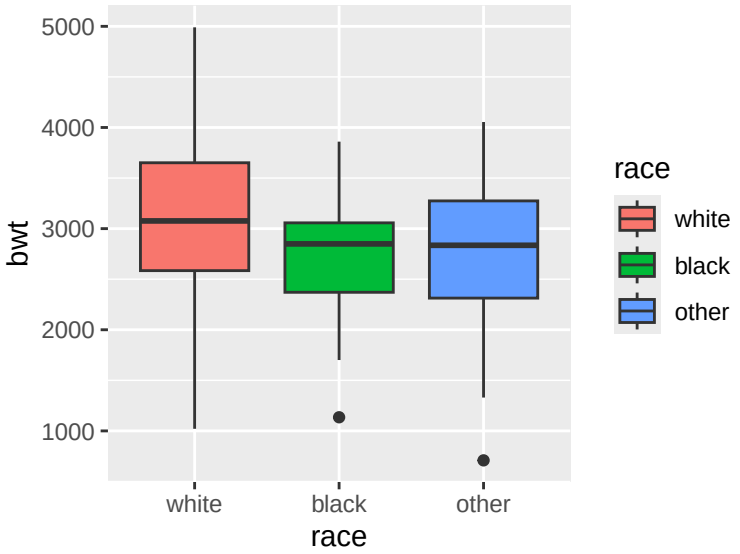
Khi ta muốn so sánh sự khác biệt về giá trị của một biến định lượng, trong các nhóm khác nhau - được xác định bởi một biến định tính, ta có thể sử dụng biểu đồ boxplot. Cú pháp lệnh vẽ là tương tự, tuy nhiên ta sẽ cần thêm `x` bên trong đối số `aes(...)` để xác định biến định tính (phải là được định dạng nhân tố - *factor*). Ví dụ, ta muốn so sánh cân nặng của trẻ sơ sinh, `bwt`, theo ba nhóm chủng tộc của người mẹ (`race`):

```
ggplot(data = data_birth, aes(x = race, y = bwt)) +
  geom_boxplot()
```



Ta có thể tô màu cho các hộp theo từng nhóm, với `fill = x`, `x` là tên của biến định tính xác định nhóm:

```
ggplot(data = data_birth, aes(x = race, y = bwt, fill = race)) +
  geom_boxplot()
```



Chú ý: Để bật tắt chú thích của biểu đồ, ta có thể sử dụng đối số `legend.position = "none"` bên trong hàm `theme()`. Xem trang trợ giúp của hàm `theme()` để biết thêm các đối số khác hiệu chỉnh bố cục của biểu đồ.

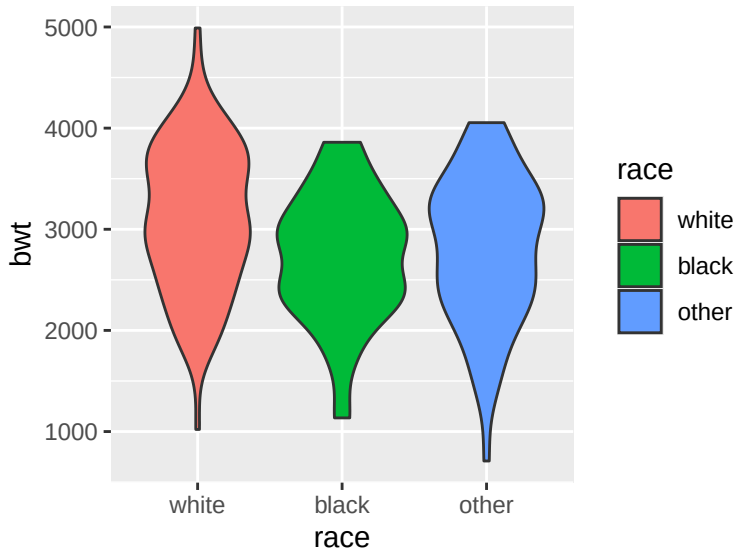
Bài tập 9:

- Tìm hiểu hàm `facet_wrap()`, và áp dụng để chia biểu đồ hộp đã vẽ ở trên thành hai biểu đồ con tương ứng trạng thái hút thuốc của người mẹ. Ta có thể nhận xét được gì?
- Vẽ lại biểu đồ boxplot cho biến `bwt` theo `race`, nhưng dùng thêm tham số `fill = smoke`; so sánh kết quả thu được ở câu trên, khi sử dụng `facet_wrap()`.
- Tìm hiểu cách sử dụng hàm `cut()` để tạo mới một biến định tính miêu tả nhóm tuổi của mẹ dựa trên dữ liệu `age`; vẽ biểu đồ boxplot miêu tả cân nặng của trẻ theo nhóm tuổi của mẹ.

- Lập lại biểu đồ boxplot trên, nhưng có thêm nhân tố trạng thái hút thuốc của mẹ;
- Dùng biểu đồ boxplot để miêu tả thay đổi cân nặng của trẻ sơ sinh theo 3 biến định tính: nhóm tuổi của mẹ, chủng tộc của mẹ, trạng thái hút thuốc.
- Nếu thêm `coord_flip()` thì ta thu được gì?
- Tìm hiểu hàm `coord_trans()`. Áp dụng để tạo biểu đồ hộp cho `log()`, `sqrt()` của cân nặng.

Bên cạnh biểu đồ hộp, có một biến thể khác của nó, với tên gọi là biểu đồ violin cũng được sử dụng trong việc mô tả mối liên hệ giữa 1 biến định lượng và 1 biến định tính. Trong R, biểu đồ violin được thực hiện bởi hàm `geom_violin()`:

```
ggplot(data = data_birth, aes(x = race, y = bwt, fill = race)) +
  geom_violin()
```



Biểu đồ này gồm hai nửa đối xứng trục, mỗi nủa tương ứng với ước lượng hàm mật độ xác suất bằng phương pháp Kernel (với bandwidth được lựa chọn bằng `ndr0`, và hàm Kernel Gaussian). Do đó, biểu đồ violin cho ta góc nhìn về tính chất phân phối của dữ liệu định lượng theo từng nhóm.

Bài tập 10:

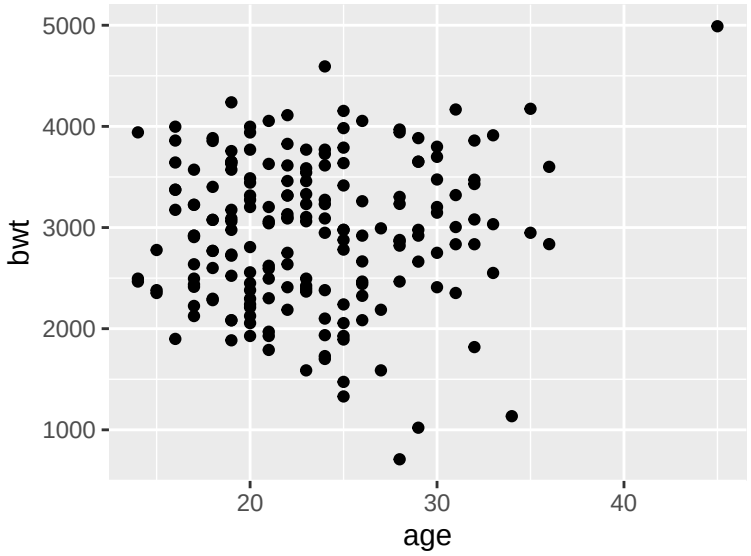
- Thực hiện lại các yêu cầu trong bài tập 9 đối với biểu đồ violin.
- Ghép đoạn lệnh vẽ biểu đồ violin với `geom_boxplot(width = 0.1)`

2.4 Biểu diễn mối liên hệ của 2 biến định lượng

Biểu đồ phân tán thường được sử dụng để mô tả sự tương quan (mối liên hệ) giữa hai biến định lượng, hoặc cũng có thể là sự tương quan giữa hai biến định lượng và một biến định tính.

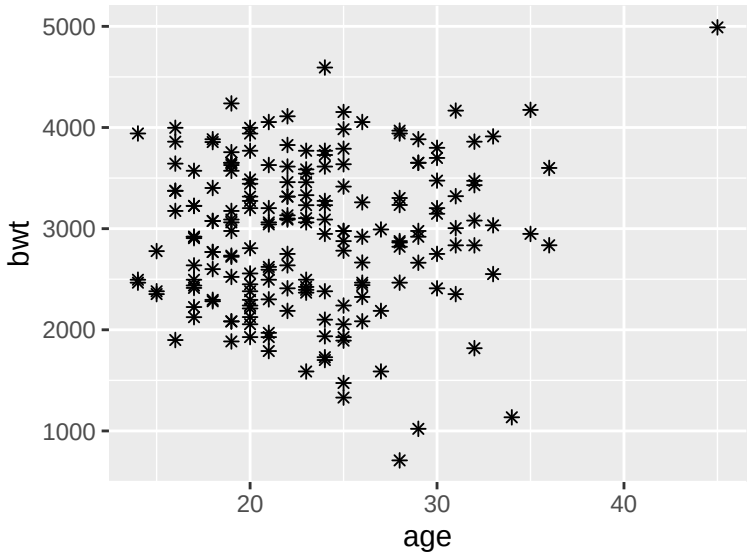
Để vẽ biểu đồ phân tán (*scatter plot*) với thư viện `ggplot2`, ta sử dụng hàm `geom_point()`. Cú pháp lệnh vẽ sẽ tương tự các phần trên. Ví dụ, ta muốn vẽ biểu đồ phân tán biểu diễn sự thay đổi của cân nặng của trẻ sơ sinh (`bwt`) theo tuổi của mẹ (`age`):

```
ggplot(data = data_birth, aes(x = age, y = bwt)) +
  geom_point()
```



Theo mặc định, hàm `geom_point()` sẽ vẽ các điểm dạng hình tròn màu đen, tuy nhiên ta có thể đổi dạng của điểm bằng đối số `shape` bằng các giá trị từ 1 tới 25. Ví dụ:

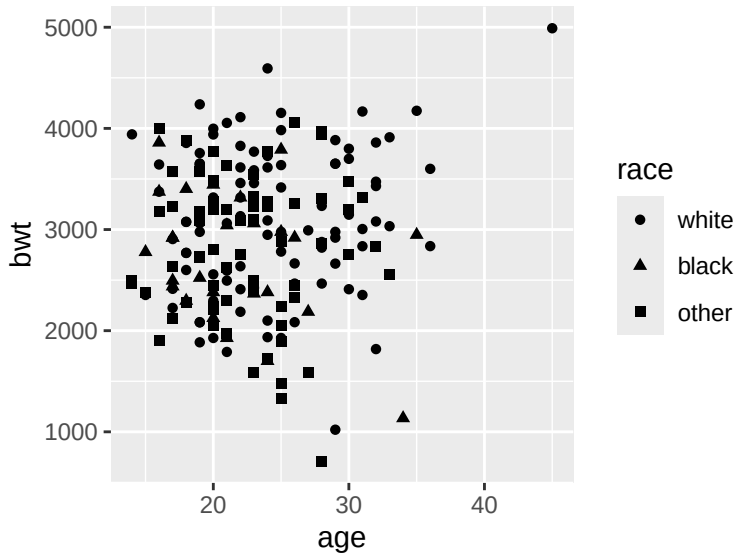
```
ggplot(data = data_birth, aes(x = age, y = bwt)) +
  geom_point(shape = 8)
```



Ngoài ra, ta cũng có thể đổi màu của các điểm bằng đối số `color`.

Trong trường hợp, ta muốn khám phá mối liên hệ của cân nặng của trẻ sơ sinh và độ tuổi, theo nhóm chủng tộc của mẹ (một biến định tính), ta có thể sử dụng `shape = race` nhằm tạo ra các kiểu hình khác nhau cho các điểm dữ liệu dựa trên thông tin nhóm chủng tộc:

```
ggplot(data = data_birth, aes(x = age, y = bwt, shape = race)) +
  geom_point()
```



Có thể dùng thêm màu sắc, hoặc hàm `facet_wrap()` nếu ta muốn làm với nhiều hơn một biến định tính.

Bài tập 11: thực hiện các thao tác sau

- vẽ lại biểu đồ phân tán cho biến `bwt` theo `age`, thay đổi tên các trục số, hiệu chỉnh màu, hình dạng của điểm và hình nền, sau đó, hãy thử với log của cân nặng;
- ghép nối đoạn lệnh vẽ biểu đồ cột phía trên với hàm `facet_wrap(~ smoke)` để tách thành hai biểu đồ con theo trạng thái hút thuốc của người mẹ; ta có thể nhận xét gì dựa trên biểu đồ về vẽ;
- vẽ lại biểu đồ phân tán cho biến `bwt` theo `age`, nhưng dùng thêm tham số `color = smoke`; so sánh kết quả thu được ở câu trên, khi sử dụng `facet_wrap()`.
- dùng biểu đồ boxplot để miêu tả thay đổi cân nặng của trẻ sơ sinh theo 3 biến định: tuổi của mẹ, chủng tộc của mẹ, trạng thái hút thuốc.

2.5 Biểu diễn mối liên hệ của nhiều biến định lượng

Để biểu diễn mối liên hệ hay sự tương quan giữa nhiều biến định lượng, ta dùng biểu đồ hệ số tương quan. Nhắc lại rằng, ta có hai loại tương quan cơ bản:

- tương quan tuyến tính Pearson;
- tương quan hạng đơn điệu Spearman.

Biểu đồ hệ số tương quan được thực hiện bởi hàm `corrplot()` được cung cấp bởi thư viện `corrplot`:

```
corrplot(corr, ...)
```

trong đó `corr` là ma trận hệ số tương quan, được tính bởi hàm `cor()`:

```
cor(x, use = "everything", method = c("pearson", "kendall", "spearman"))
```

trong đó,

- `x` là ma trận chứa các cột là các biến cần tính hệ số tương quan (chú ý tất cả các biến phải là biến định lượng);
- `use` là lựa chọn được sử dụng khi ma trận dữ liệu có dữ liệu khuyết, ta có các lựa chọn sau: `"everything"`, `"all.obs"`, `"complete.obs"`, `"na.or.complete"`, hoặc `"pairwise.complete.obs"`;

- `method` là tên của phương pháp dùng để tính hệ số tương quan, ta cần lựa chọn 1 trong số 3 phương pháp được cung cấp.

```
library(corrplot)
```

Ta sử dụng dữ liệu `mtcars` được cung cấp sẵn trong `R`. Dữ liệu ghi chép lại thông số kỹ thuật của 32 dòng xe moto tại Mỹ.

```
data("mtcars")
glimpse(mtcars)
```

```
## Rows: 32
## Columns: 11
## $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, 16.4~
## $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, 8, 8~
## $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 167.6, ~
## $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180, 205~
## $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92, 3.07~
## $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.440, ~
## $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18.30, ~
## $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0~
## $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0~
## $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3~
## $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2, 2, 4~
```

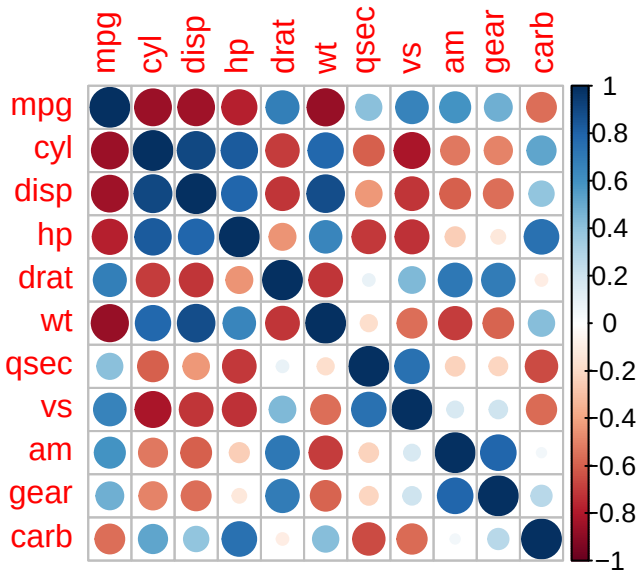
Ta tính ma trận hệ số tương quan tuyến tính

```
cor_mtcars <- cor(mtcars, method = "pearson")
cor_mtcars
```

```
##             mpg        cyl       disp         hp        drat         wt        qsec
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594  0.41868403
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958 -0.59124207
## disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
## vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157  0.74453544
## am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953 -0.22986086
## gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870 -0.21268223
## carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059 -0.65624923
##              vs          am       gear        carb
## mpg   0.6640389  0.59983243  0.4802848 -0.55092507
## cyl  -0.8108118 -0.52260705 -0.4926866  0.52698829
## disp -0.7104159 -0.59122704 -0.5555692  0.39497686
## hp   -0.7230967 -0.24320426 -0.1257043  0.74981247
## drat  0.4402785  0.71271113  0.6996101 -0.09078980
## wt   -0.5549157 -0.69249526 -0.5832870  0.42760594
## qsec  0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs    1.0000000  0.16834512  0.2060233 -0.56960714
## am    0.1683451  1.00000000  0.7940588  0.05753435
## gear  0.2060233  0.79405876  1.0000000  0.27407284
## carb -0.5696071  0.05753435  0.2740728  1.00000000
```

Chú ý, do dữ liệu `mtcars` chứa toàn bộ các biến định lượng, nên ta có thể nhập `mtcars` vào trong hàm `cor()`. Biểu đồ hệ số tương quan được vẽ như sau:

```
corrplot(cor_mtcars)
```



Bài tập 12: Thử các dạng khác nhau của đồ thị bằng đối số `method = ...` (tìm hiểu các lựa chọn trong help page của hàm `corrplot()`).

Bài tập 13: Vẽ biểu đồ hệ số tương quan hạng đơn điệu Spearman cho các biến trong dữ liệu `mtcars`.

Bài tập 14: Vẽ biểu đồ hệ số tương quan cho các biến trong dữ liệu `Advertising.csv`.

3 Xuất file hình

Hình vẽ biểu đồ trong `R/Rstudio`, vào một tệp định dạng hình ảnh như .png, .jpg hoặc định dạng vector như .pdf, .eps. Có nhiều hàm khác nhau để thực hiện việc xuất/lưu trữ hình vẽ trong `R`, tuy nhiên, lựa chọn tốt nhất là các cách lưu dựa trên cairo graphics API, vì giúp lưu hình với chất lượng cao và lưu được các tiêu đề bằng tiếng Việt.

Để chất lượng hình lưu không bị ảnh hưởng khi thu phóng hình, ta nên dùng định dạng cuối là .pdf, định dạng này có thể dán được trong văn bản khi soạn thảo bằng latex. Tuy nhiên, nếu sử dụng word thì ta nên sử dụng định dạng cuối là .png. Cụ thể:

- `cairo_pdf(filename = "ten_hinh.pdf", width = 6, height = 5)`: dùng để xuất hình dưới dạng tệp .pdf, trong đó, `width` và `height` lần lượt là độ rộng và chiều cao của hình, được tính theo inch, ảnh hưởng tới độ thu phóng của hình;
- `png(filename = "ten_hinh.png", width = 480, height = 480, type = "cairo")`: dùng để xuất hình dưới dạng tệp .png, trong đó, `width` và `height` được tính theo pixel.
- `jpeg(filename = "ten_hinh.jpg", width = 480, height = 480, quality = 100, type = "cairo")`: dùng để xuất hình dưới dạng tệp .jpg

Cấu trúc đoạn lệnh như sau:

```
cairo_pdf(filename = "ten_hinh.pdf", width = 6, height = 5)
\\ đoạn lệnh vẽ hình
dev.off()
```

Ví dụ, với biểu đồ phân tán miêu tả sự thay đổi của cân nặng trẻ sơ sinh theo độ tuổi của người mẹ. Đối với tiêu đề hình bằng tiếng Việt, ta cần chuyển thành các đoạn code UTF-8 (sử dụng công cụ trong đường dẫn

sau https://www.vietnamesetools.com/vi/vietnamese-to-unicode). Ta có thể lưu lại theo các định dạng file khác nhau như sau:

```
cairo_pdf(filename = "bieu_do_phan_tan_1.pdf", width = 6, height = 5)
ggplot(data = data_birth, aes(x = age, y = bwt)) +
  geom_point() +
  xlab("Tuổi") + ylab("C\u00e2n nặng của trẻ sơ sinh") +
  theme_classic()
dev.off()
```

```
png(filename = "bieu_do_phan_tan_1.png", width = 480, height = 480,
    type = "cairo")
ggplot(data = data_birth, aes(x = age, y = bwt)) +
  geom_point() +
  xlab("Tuổi") + ylab("C\u00e2n nặng của trẻ sơ sinh") +
  theme_classic()
dev.off()
```

```
jpeg(filename = "bieu_do_phan_tan_1.jpg", width = 480, height = 480,
    quality = 100, type = "cairo")
ggplot(data = data_birth, aes(x = age, y = bwt)) +
  geom_point() +
  xlab("Tuổi") + ylab("C\u00e2n nặng của trẻ sơ sinh") +
  theme_classic()
dev.off()
```