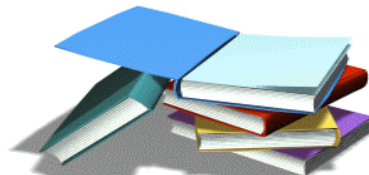
TRƯỜNG ĐẠI HỌC
KINH TẾ KỸ THUẬT CÔNG NGHIỆP
UNIVERSITY OF ECONOMIC AND TECHNICAL INDUSTRIES
UNETI

Slide Đồ án 1:

Trực quan hóa dữ liệu với R

GVHD: Trần Chí Lê

Version: 1



TÀI LIỆU HỌC TẬP

❑ Tài liệu chính:

[1] ***Tài liệu học tập Đồ án 1*****, Trường Đại học Kinh tế-Kỹ thuật Công nghiệp.**

[2] ***Phân tích dữ liệu với R*****, Nguyễn Văn Tuấn, NXB thành phố HCM.**

❑ Tài liệu tham khảo:

[1] ***Tài liệu học tập Thống kê Toán học cho ngành Khoa học dữ liệu*****, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp.**

[2] Eric Pimpler (2017), ***Data Visualization and Exploration with R*****.**

PHƯƠNG THỨC ĐÁNH GIÁ VÀ KIỂM TRA

- **Điểm Đồ án môn học:** là trung bình cộng của các điểm sau:

1. Điểm chuyên cần (*phần mềm tự động tính*)
2. Điểm giữa kỳ (*đề cương đồ án+file dữ liệu*)
3. Điểm cuối kỳ (*sản phẩm + báo cáo*)

NỘI DUNG

Bài 1: Các kỹ thuật thăm dò và trực quan hóa dữ liệu với R

Bài 2: Chọn bài toán trong các lĩnh vực tự nhiên và xã hội để phân tích dữ liệu

Bài 3: Mô hình hóa bài toán

Bài 4: Giải quyết bài toán

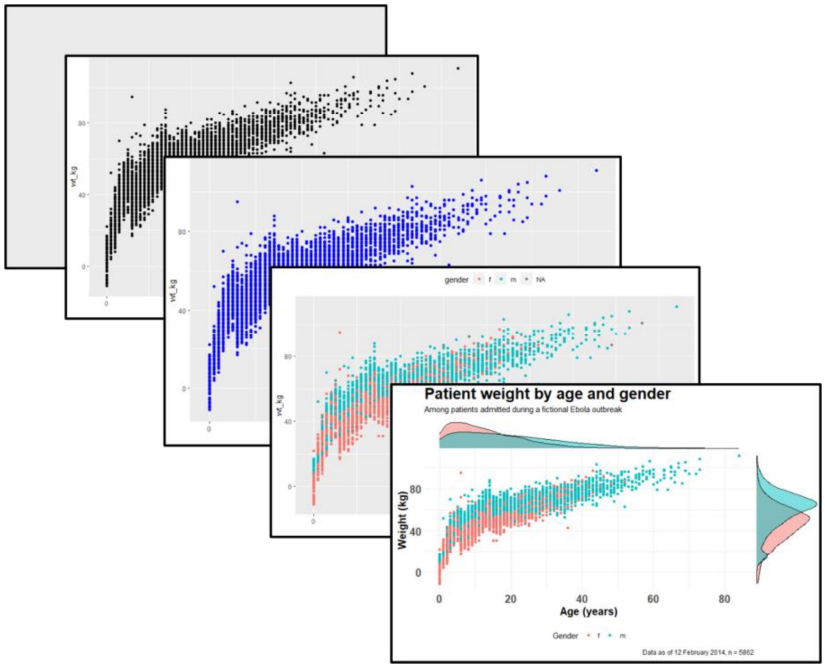
Bài 5: Tổng kết và báo cáo

Bài 1
CÁC KỸ THUẬT THĂM DÒ VÀ TRỰC QUAN HÓA DỮ LIỆU CƠ BẢN TRONG R

Tóm tắt kiến thức: Bài này giới thiệu các kỹ thuật phân tích thống kê (ước lượng, kiểm định các tham số của các số liệu mẫu dựa trên quy luật phân phối tương ứng) và kỹ thuật xây dựng biểu đồ cho từng đối tượng dữ liệu (về độ lớn dữ liệu, dữ liệu biểu thị dạng phần trăm, về quy luật phân phối của dữ liệu và mối tương quan giữa các nhóm biến trong dữ liệu), (xem [1,2,14,15]).

1.1. THĂM DÒ DỮ LIỆU BẰNG BIỂU ĐỒ

1.1.1. Biểu đồ với package ggplot2



Hình 1.1: Minh họa qui trình thêm các lớp hàm phân tích trong ggplot.

Để minh họa cho việc sử dụng ggplot chúng ta sẽ làm việc trên một dữ liệu tích hợp cùng gói package *ggplot2*, đó là dữ liệu *mpg* chứa các quan sát được Cơ quan Bảo vệ Môi trường Hoa Kỳ thu thập trên 38 mẫu ô tô với 233 quan sát và 11 biến. (trong thư viện ggplot2 thuộc R gõ: ?mpg để biết chi tiết về nguồn gốc dữ liệu).

```
mpg
#> # A tibble: 234 × 11
#>   manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
#>   <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
#> 1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa…
#> 2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa…
#> 3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa…
#> 4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa…
```

```
#> 5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa…
#> 6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa…
#> # ℹ 228 more rows
```

Bảng 1.1: Dữ liệu quan sát về các mẫu ô tô, (nguồn: ggplot2).

a) Cú pháp cơ bản

Chúng ta có thể minh họa cú pháp cơ bản như sau:

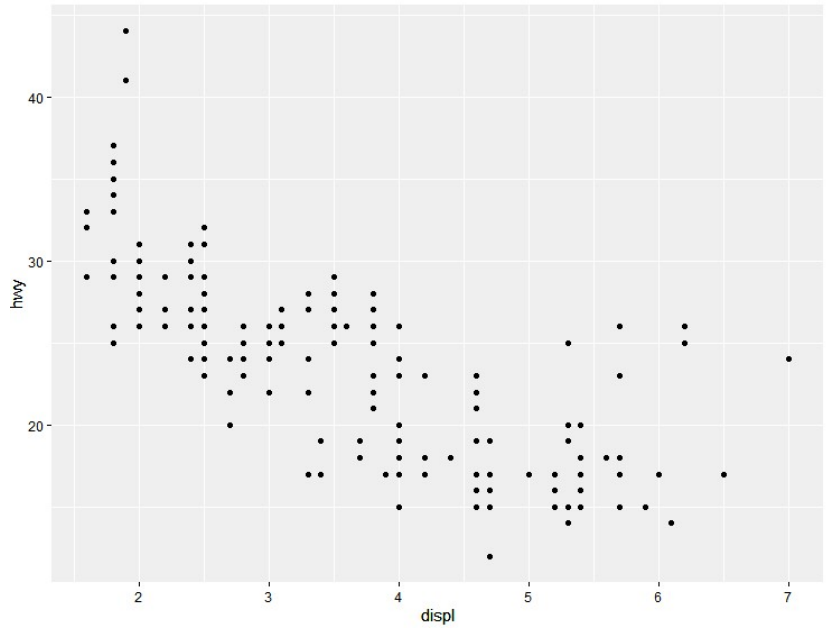
```
ggplot(data = my_data)+                    # sử dụng dữ liệu "my_data"
  geom_yyy(                               # thêm một lớp các hàm-hình biểu đồ
    mapping = aes(x = col1, y = col2),    # gán dữ liệu tới các trục
    color = "red")+                       # thêm một số đặc điểm khác (như màu sắc)
  labs()+                                 # thêm tiêu đề, nhãn, bảng số,..
  theme()                                 # điều chỉnh cỡ chữ, màu sắc, phông chữ
```

b) Gán các biến dữ liệu cho biểu đồ

Hầu hết các hàm-hình geom phải được cho biết cái gì được sử dụng để vẽ biểu đồ, vì vậy chúng ta phải cung cấp cách map (gán) các biến số trong dữ liệu tới các thành phần của biểu đồ như là các trục, màu đối tượng, kích thước đối tượng, v.v. Đối với hầu hết các geoms, các thành phần thiết yếu phải được gán tới các cột trong dữ liệu là trục x, và (nếu cần) là trục y.

Ví dụ 1.1. Trong lệnh ggplot() dưới đây, dữ liệu được thiết lập là bộ dữ liệu mpg. Trong đó đối số mapping = aes(), cột displ được gán cho trục x, và cột hwy được gán cho trục y.

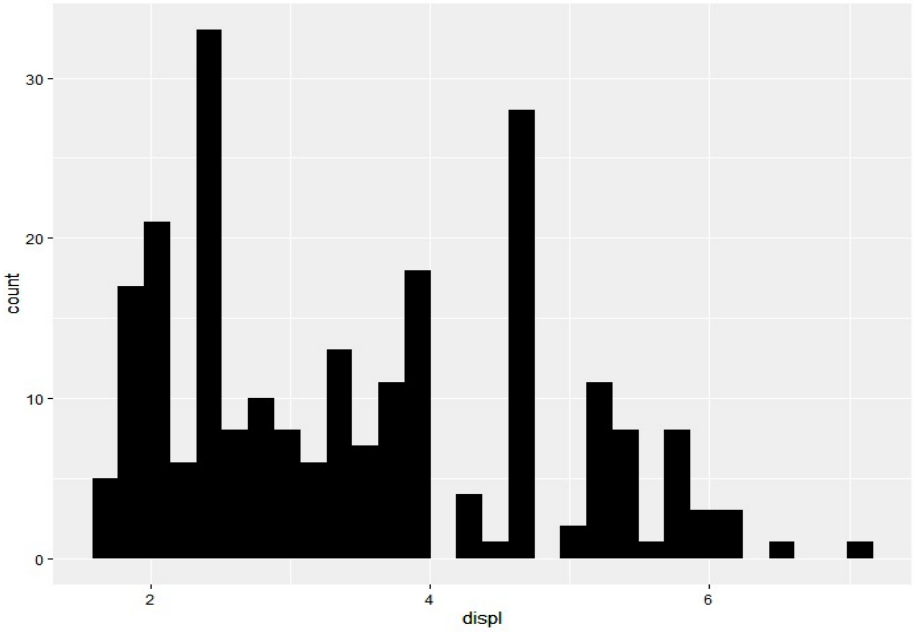
```
ggplot(data = my_data)+
geom_poit(mapping = aes(x = displ, y = hwy))
# kết quả hiển thị:
```



Hình 1.2: Cách gán biến số vào các trục.

Ví dụ 1.2. Lệnh sau tương tự Ví dụ 1.1, chỉ có một sự khác biệt nhỏ về cách mapping và hàm geom. Hàm geom_histogram() chỉ yêu cầu gán cột cho trục x, bởi vì trục số lượng y được tạo ra một cách tự động bằng số phần tử của mỗi biến x.

```
ggplot(data = mpg, mapping = aes(x = displ))+
  geom_histogram()
# kết quả hiển thị:
```



Hình 1.3: Cách gán biến vào lớp hàm geom_histogram.

c) Tính thẩm mỹ trong biểu đồ

Tính thẩm mỹ trong biểu đồ có thể là màu sắc, kích thước, độ trong suốt, vị trí, v.v. của dữ liệu được vẽ. Không phải tất cả các *geoms* sẽ có các tùy chọn về tính thẩm mỹ, trang trí giống nhau, nhưng một số tùy chọn được áp dụng với phần lớn các *geoms*. Dưới đây là một số trang trí hay gặp:

- shape = Hiển thị một điểm với hàm geom_point() dưới dạng dấu chấm, ngôi sao, hình tam giác hoặc hình vuông,…
- fill = Màu sắc bên trong (vd: của cột hoặc boxplot).
- color = Đường bên ngoài của cột, boxplot, v.v., hoặc màu của điểm nếu sử dụng hàm geom_point().
- size = Kích thước (vd: độ dày của đường, kích thước của điểm).
- alpha = Độ trong suốt (1 = bình thường, 0 = vô hình).
- binwidth = Độ rộng các bins trong biểu đồ histogram.
- width = Độ rộng của các cột trong "biểu đồ cột".
- linetype = Kiểu của đường (vd: liền, nét đứt, chấm chấm).

Trang trí của đối tượng biểu đồ có thể được gán giá trị theo hai cách: Gán một giá trị tĩnh (vd: color = "blue") để áp dụng cho tất cả các quan sát được vẽ biểu đồ hoặc gán cho từng

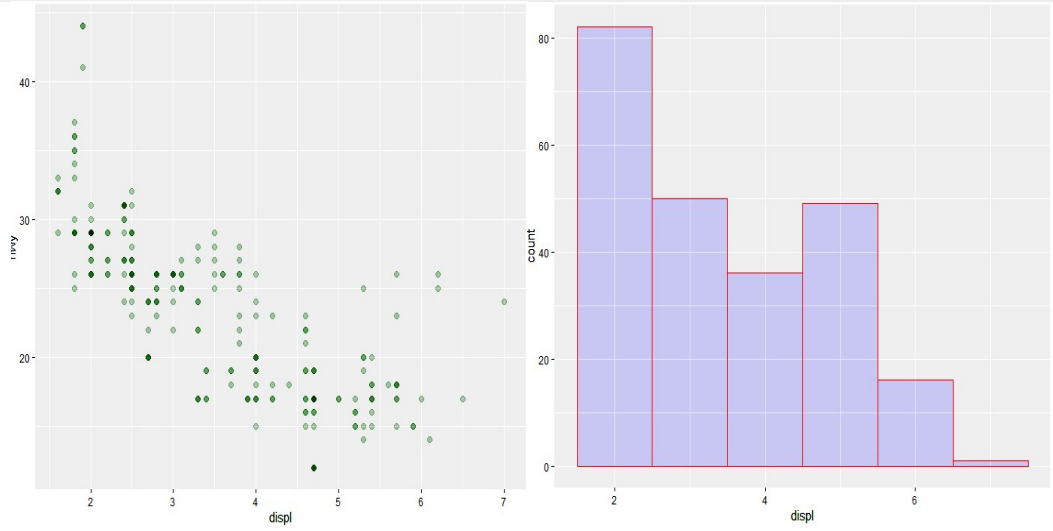
biến của dữ liệu (vd: color = hospital) để hiển thị từng quan sát phụ thuộc vào giá trị của nó trong biến đó.

- ***Trang trí với một giá trị tĩnh***

Nếu muốn yếu tố trang trí cho đối tượng biểu đồ là tĩnh, nghĩa là - giống nhau đối với mọi quan sát trong dữ liệu, chúng ta gán nó *bên trong* geom nhưng *ở bên ngoài* đối số mapping = aes() . Các phép gán này có thể ví dụ như: size = 1 hoặc color = "blue".

Ví dụ 1.3. Xét bộ dữ liệu mpg, với phép gán giá trị tĩnh về màu sắc

```
# biểu đồ vô hướng
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+    # gán dữ liệu và các trục
  geom_point(color = "darkgreen", size = 2, alpha = 0.2)  # hàm tạo điểm
# biểu đồ phân bố
ggplot(data = mpg, mapping = aes(x = displ))+             # gán dữ liệu và các trục
  geom_histogram(                                         # hàm phân bố
    binwidth = 1,                                         # độ rộng cột
    color = "red",                                        # đường màu
    fill = "blue",                                        # màu tô bên trong
    alpha = 0.1)                                          # độ trong
# kết quả hiện thị
```



Hình 1.4: Phép gán tĩnh.

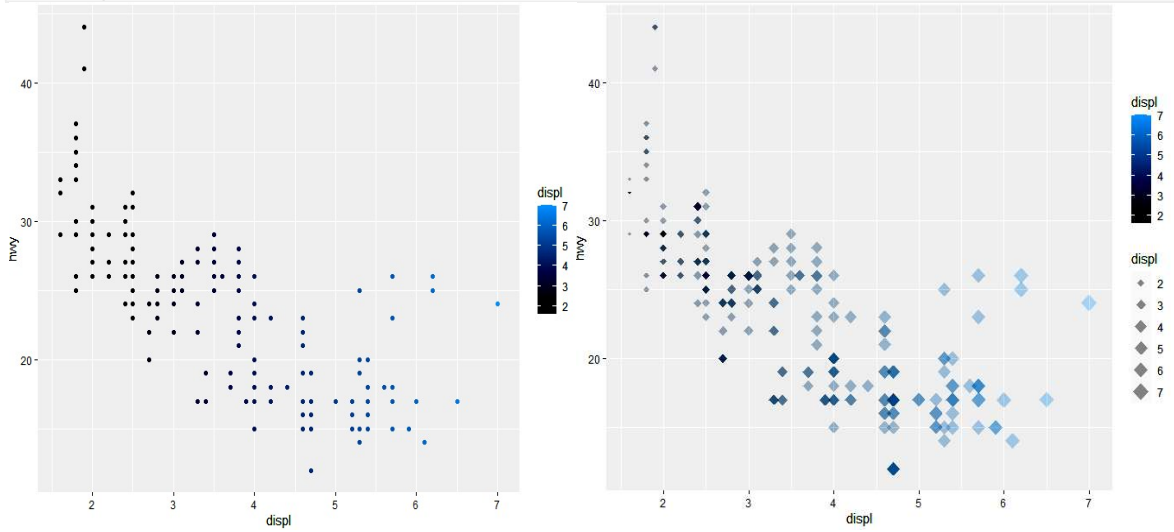
- ***Trang trí theo giá trị của từng biến***

Để thực hiện được điều này, chúng ta gán yếu tố trang trí của biểu đồ với một biến (không trong dấu ngoặc kép). Điều này phải được thực hiện *bên trong một hàm* mapping = aes()

Ví dụ 1.4. Xét bộ dữ liệu mpg, với phép trang trí theo biến x = displ bởi màu sắc-color hoặc kích cỡ-size.

```
# biểu đồ vô hướng 1
ggplot(data = mpg, mapping = aes( x = displ, y = hwy, color = displ))+
  geom_point()
```

```
# biểu đồ vô hướng 2
ggplot(data = mpg, mapping = aes( x = displ, y = hwy,color = displ,size = displ))+
  geom_point(shape = "diamond", alpha = 0.3)
# kết quả hiển thị:
```



Hình 1.5: Trang trí theo biến.

Trong biểu đồ đầu tiên, yếu tố thẩm mỹ color (của mỗi điểm) được gán cho biến displ - và thang đo liên tục được xuất hiện dưới dạng chú thích. Trong biểu đồ thứ hai, hai yếu tố trang trí được gán cho biến displ với hai yếu tố thẩm mỹ là color và size, trong khi shape và alpha được gán cho các giá trị tĩnh bên ngoài đối số mapping = aes().

Nhận xét 1: Các phép gán trục luôn được gán cho các biến trong dữ liệu (không phải cho các giá trị tĩnh) và điều này luôn được thực hiện với mapping = aes(). Điều quan trọng là phải theo dõi các lớp-geom của biểu đồ và các đối tượng thẩm mỹ khi vẽ các biểu đồ phức tạp - ví dụ biểu đồ được cấu thành từ nhiều geoms.

Nhận xét 2: Việc gán các yếu tố trang trí bên trong đối số mapping = aes() có thể được viết ở một số chỗ trong các lệnh vẽ biểu đồ và thậm chí có thể được viết nhiều lần. Nó có thể được viết trong lệnh ggplot() trên cùng, hoặc cho từng geom riêng lẻ bên dưới. Các kiểu viết bao gồm:

- Các phép gán được thực hiện ở lệnh ggplot() trên cùng sẽ được mặc định kế thừa ở bất kỳ các geom bên dưới, giống như cách mà x = và y = được kế thừa.
- Các phép gán được thực hiện trong một geom chỉ áp dụng cho geom đó.
- Tương tự, data = được chỉ định cho lệnh ggplot() ở trên đầu sẽ áp dụng mặc định cho tất cả các geom bên dưới.

Ví dụ 1.6. Mỗi lệnh sau sẽ tạo ra cùng một biểu đồ giống nhau:

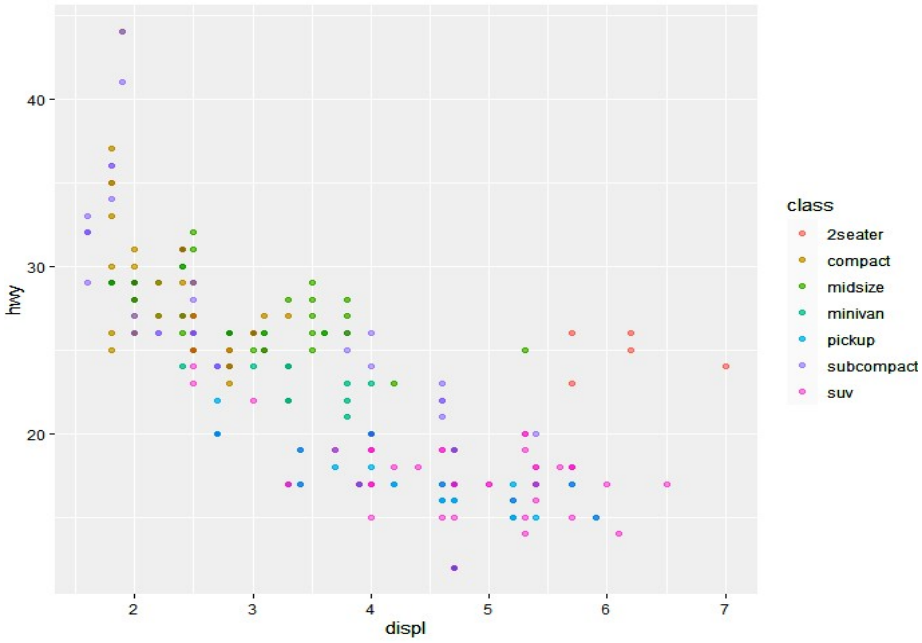
```
# Mẫu thứ nhất
ggplot(data = mpg, mapping = aes(x = displ))+
  geom_histogram()
# Mẫu thứ hai
```

```
ggplot(data = mpg)+
  geom_histogram(mapping = aes(x = displ))
# Mẫu thứ ba
ggplot()+
  geom_histogram(data = mpg, mapping = aes(x = displ))
```

- ***Trang trí theo nhóm đối tượng***

Ví dụ 1.7. Xét lại dữ liệu mpg, để có thông tin chi tiết hơn về dữ liệu, chúng ta có thể sử dụng màu sắc để phân biệt tới biến class hiển thị kiểu dáng của từng chiếc xe. Chúng ta sẽ đặt mapping = aes(color = "class") khi đó một chú giải tự động xuất hiện . Phép gán này có thể được thực hiện bên trong mapping = aes() ở lệnh ggplot() đầu tiên (và được thừa kế bởi các geom), hoặc nó thể được đặt trong một mapping = aes() riêng biệt bên trong geom. Cả hai cách tiếp cận được trình bày dưới đây:

```
# cách tiếp cận 1
ggplot(data = mpg,mapping = aes(x = displ, y = hwy, color = class))+
  geom_point(alpha = 0.5)
# cách tiếp cận 2
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(mapping = aes(color = class),alpha = 0.5)
# kết quả hiển thị:
```



Hình 1.6: Trang trí màu sắc theo nhóm.

Lưu ý rằng tùy thuộc vào loại geom sử dụng, chúng ta sẽ cần sử dụng các đối số khác nhau để trang trí cho nhóm đối tượng. Đối với geom_point(), ta thường sử dụng các tham số như color , shape hoặc size. Trong khi đó đối với geom_bar(), ta thường sử dụng tham số fill. Điều này chỉ phụ thuộc vào loại geom và yếu tố trang trí nào mà chúng ta muốn thể hiện sự phân nhóm.

d) Gán nhãn cho biểu đồ

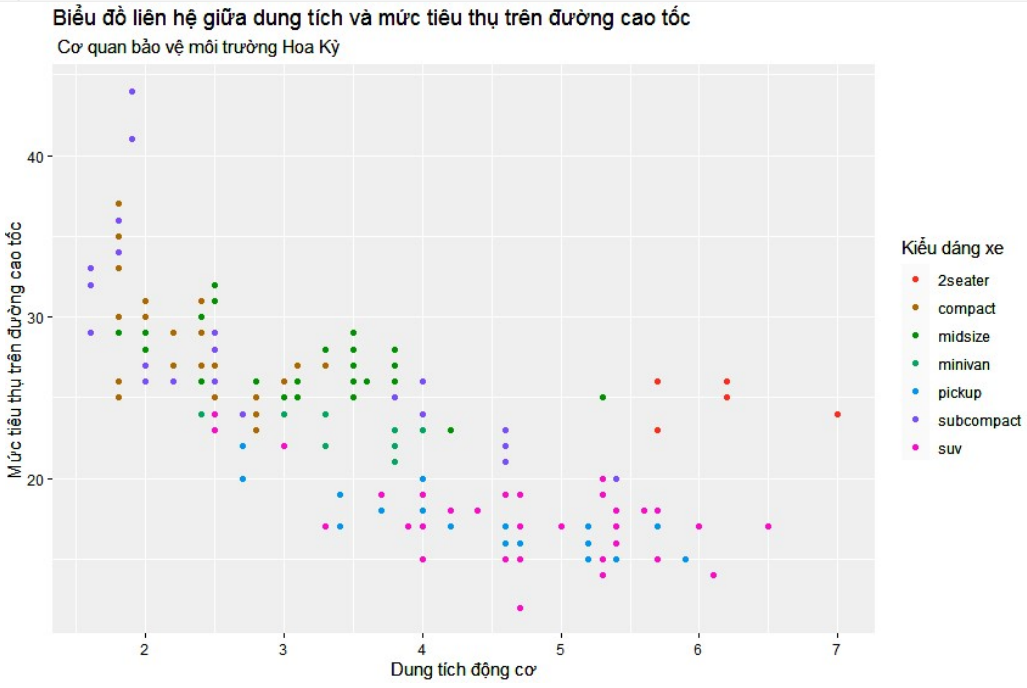
Việc đặt tên cho tiêu đề biểu đồ, tên các biến trên trục, các chú thích là công việc không thể thiếu khi vẽ biểu đồ, và việc này được thực hiện với hàm labs() bằng cách thêm dấu + như cách chúng ta thêm các geoms.

Bên trong hàm labs(), cung cấp các chuỗi ký tự cho các đối số sau:

- x = và y = Tiêu đề trục x và trục y (nhãn).
- title = Tiêu đề chính của biểu đồ.
- subtitle = Tiêu đề phụ của biểu đồ, nhỏ hơn và đặt bên dưới tiêu đề chính.
- caption = Chú thích của biểu đồ, mặc định ở góc phải dưới.

Ví dụ 1.8. Ví dụ dưới đây là biểu đồ chúng ta đã tạo ở Ví dụ 1.7 nhưng có thêm các nhãn:

```
Bieu_do1<-ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy,color=class))+
  labs(title = "Biểu đồ liên hệ giữa dung tích và mức tiêu thụ trên đường cao tốc",
     subtitle = " Cơ quan bảo vệ môi trường Hoa Kỳ",x = "Dung tích động cơ",y = "Mức
tiêu thụ trên đường cao tốc", color = "Kiểu dáng xe")
# in biểu đồ
Bieu_do1
# Kết quả hiển thị:
```



Hình 1.7: Gán nhãn cho biểu đồ.

e)Căn chỉnh trong biểu đồ

Việc căn chỉnh màu nền của biểu đồ, sự xuất hiện/biến mất của đường lưới, cũng như phông chữ/cỡ chữ/màu sắc/căn lề của văn bản (tiêu đề chính, tiêu đề phụ, Chú thích, chữ trên các trục…). được thực hiện theo hai cách: Căn chỉnh theo mặc định sẵn có và căn chỉnh cá nhân đơn lẻ

- ***Căn chỉnh theo mặc định***

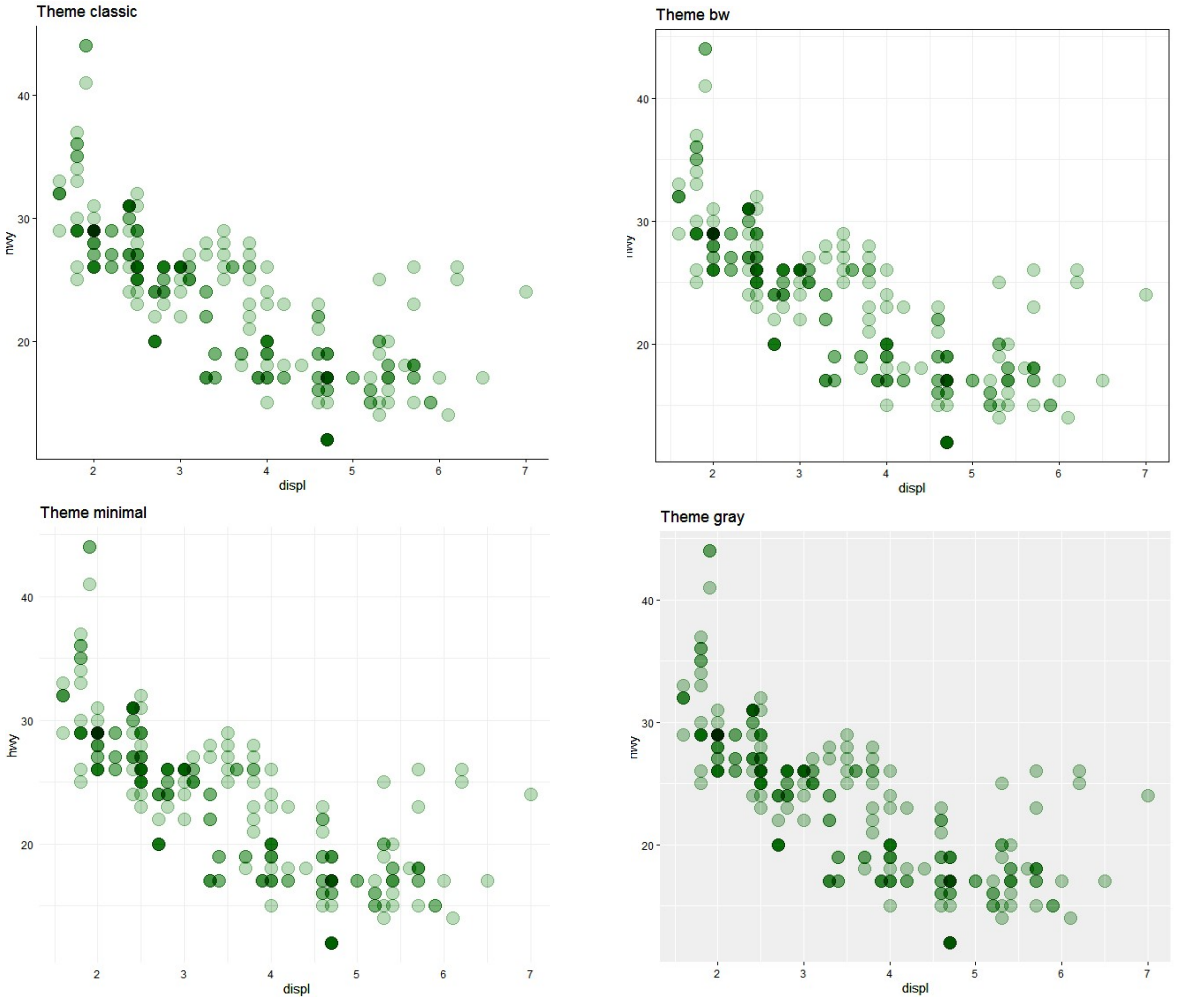
  Căn chỉnh theo mặc định, tức là chúng ta sử dụng căn chỉnh theo một *chủ đề hoàn chỉnh* bằng hàm theme_() để điều chỉnh toàn bộ các thành phần biểu đồ. Cách căn chỉnh này khá đơn giản, chúng ta có thể sử dụng một số hàm chủ đề hoàn chỉnh bên dưới đây.

  - `theme_gray()`: Chủ đề ggplot2 đặc trưng với nền màu xám và đường lưới màu trắng, được thiết kế để đưa dữ liệu về phía trước nhưng vẫn giúp việc so sánh trở nên dễ dàng.
  - `theme_bw()`: Chủ đề ggplot2 tối trên ánh sáng cổ điển. Có thể hoạt động tốt hơn cho bài thuyết trình trình chiếu bằng máy chiếu.
  - `theme_linedraw()`: Một chủ đề chỉ có các đường màu đen có chiều rộng khác nhau trên nền trắng, gợi nhớ đến một bản vẽ đường. Phục vụ mục đích tương tự như theme_bw(). Lưu ý rằng chủ đề này có một số dòng rất mỏng (<< 1 pt) khi in ấn rất dễ mất hình ảnh.
  - `theme_light()`: Một chủ đề tương tự như theme_linedraw() nhưng có các đường và trục màu xám nhạt, để hướng sự chú ý nhiều hơn tới dữ liệu.
  - `theme_dark()`: Tương tự màu tối của theme_light(), với kích thước dòng tương tự nhưng nền tối, hữu ích để làm nổi bật những đường màu mảnh.
  - `theme_minimal()`: Một chủ đề tối giản không có chú thích nền.
  - `theme_classic()`: Một chủ đề có giao diện cổ điển với các đường trục x và y và không có đường lưới.
  - `theme_void()`: Một chủ đề hoàn toàn trống rỗng.
  - `theme_test()`: Một chủ đề cho bài kiểm tra đơn vị trực quan. Lý tưởng nhất là nó không bao giờ thay đổi ngoại trừ cho các tính năng mới.

Ví dụ 1.9. Ví dụ dưới đây minh họa một vài căn chỉnh theo chủ đề mặc định:

```
# căn chỉnh theo chủ đề Theme classic
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(color = "darkgreen", size = 0.5, alpha = 0.2)+
  labs(title = "Theme classic")+
  theme_classic()
# căn chỉnh theo chủ đề Theme bw
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(color = "darkgreen", size = 0.5, alpha = 0.2)+
  labs(title = "Theme bw")+
  theme_bw()
# căn chỉnh theo chủ đề Theme minimal
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(color = "darkgreen", size = 0.5, alpha = 0.2)+
  labs(title = "Theme minimal")+
```

```
  theme_minimal()
# căn chỉnh theo chủ đề Theme gray
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(color = "darkgreen", size = 0.5, alpha = 0.2)+
  labs(title = "Theme gray")+
  theme_gray()
# kết quả hiển thị:
```



Hình 1.8: Căn chỉnh theo chủ đề mặc định.

- ***Căn chỉnh cá nhân đơn lẻ***

Hàm `theme()` có thể nhận một số lượng lớn các đối số, mỗi đối số sẽ chỉnh sửa một khía cạnh rất cụ thể của biểu đồ. Chúng ta sẽ không trình bày tất cả các đối số, nhưng sẽ tập trung mô tả công thức chung cho chúng và chỉ cách tìm tên đối số khi cần. Cú pháp cơ bản là:

  - Bên trong hàm `theme()`, hãy viết tên đối số cho phần tử biểu đồ mà ta muốn chỉnh sửa, chẳng hạn như `plot.title =`.
  - Cung cấp một hàm `element_()` tới đối số.

- Thường sử dụng nhất là `element_text()`, một số khác bao gồm `element_rect()` chọn màu nền cho canvas, hoặc `element_blank()` để xóa các phần tử biểu đồ.
- Bên trong hàm `element_()`, xác định giá trị đối số cần gán để điều chỉnh theo ý bạn mong muốn.

Sau đây là một số đối số phổ biến của hàm `theme()`.

Đối số theme()	**Những điều chỉnh**
`plot.title = element_text()`	Tiêu đề chính
`plot.subtitle = element_text()`	Tiêu đề phụ
`plot.caption = element_text()`	Liên quan tới caption (kiểu font, màu sắc, kích cỡ, góc độ, vjust, hjust…)
`axis.title = element_text()`	Tiêu đề trục (cả trục x và y) (kích cỡ, góc độ, màu sắc…)
`axis.title.x = element_text()`	Chỉ tiêu đề trục x (sử dụng .y để chỉ áp dụng với trục y)
`axis.text = element_text()`	Văn bản trên trục (cả trục x và y)
`axis.text.x = element_text()`	Chỉ văn bản trục x (sử dụng .y để chỉ áp dụng với trục y)
`axis.ticks = element_blank()`	Loại bỏ ticks của trục
`axis.line = element_line()`	Đường trục (màu sắc, kích thước, kiểu đường: nét đứt, nét liền mảnh, v.v.)
`strip.text = element_text()`	Văn bản trong Facet strip (màu sắc, kích thước, góc độ…)
`strip.background = element_rect()`	facet strip (tô màu, màu sắc, kích thước…)

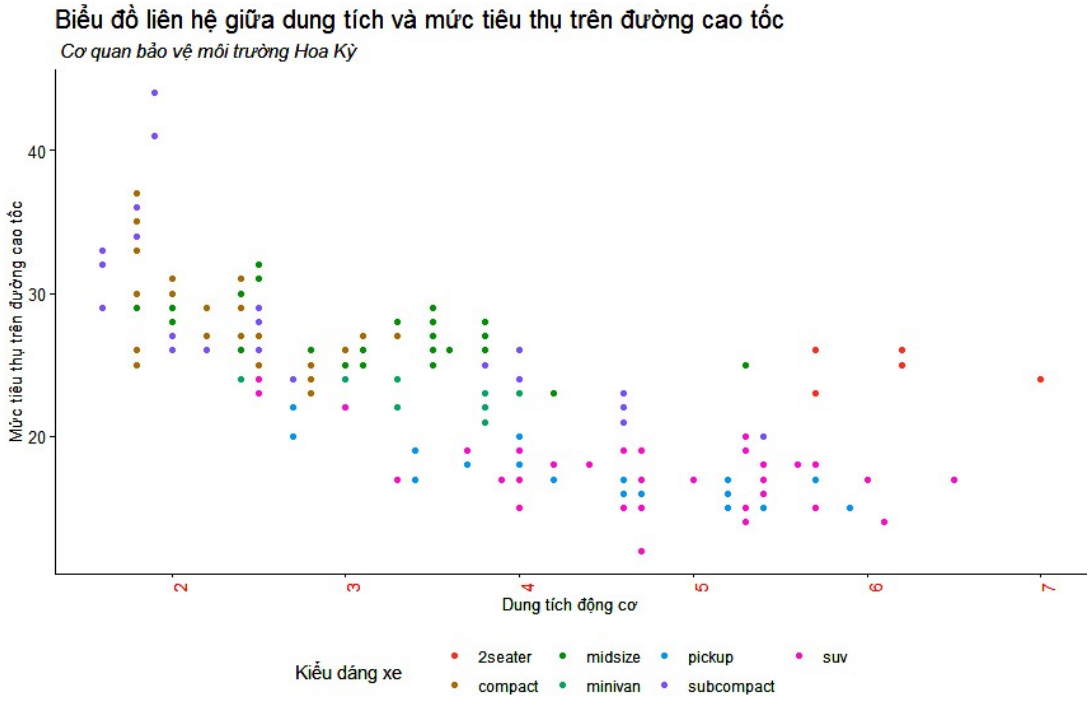
Bảng 1.2: Các đối số hay sử dụng cho việc căn chỉnh trong hàm theme.

Có một số đối số khác ít phổ biến hơn, nhưng nếu cần chúng ta có thể liệt kê ra chúng bằng cách: Chạy lệnh `theme_get()` từ *ggplot2* để in tất cả hơn 90 đối số của hàm `theme()` ra console. Hoặc nếu chúng ta muốn xóa một phần tử của biểu đồ, bạn cũng có thể làm điều đó bằng hàm `theme()`. Chỉ cần đặt `element_blank()` tới đối số để nó biến mất hoàn toàn. Đối với chú thích, thiết lập `legend.position = "none"`.

Ví dụ 1.10. Ví dụ dưới đây sử dụng biểu đồ từ file đã lưu Bieu_do1 và thêm vào một vài căn chỉnh theo tùy chọn cá nhân

```
Bieu_do1+
  theme_classic()+
  theme(legend.position = "bottom",
    plot.title = element_text(size = 15),
    plot.caption = element_text(hjust = 0),
```

```
    plot.subtitle = element_text(face = "italic"),
    axis.text.x = element_text(color = "red", size = 10, angle = 90),
    axis.text.y = element_text(size = 10),
    axis.title = element_text(size = 10))
```



Hình 1.9: Căn chỉnh biểu đồ theo tùy chọn.

Giải thích:

- `legend.position =` là đặc biệt nhất vì nó chỉ chấp nhận các giá trị đơn giản như "bottom", "top", "left", và "right". các đối số liên quan đến văn bản yêu cầu bạn đặt các chi tiết *bên trong* hàm `element_text()`.
- Cỡ chư tiêu đề với `element_text(size = 30)`.
- Căn lề caption với `element_text(hjust = 0)` (từ trái qua phải).
- Tiêu đề phụ được in nghiêng với `element_text(face = "italic")`.

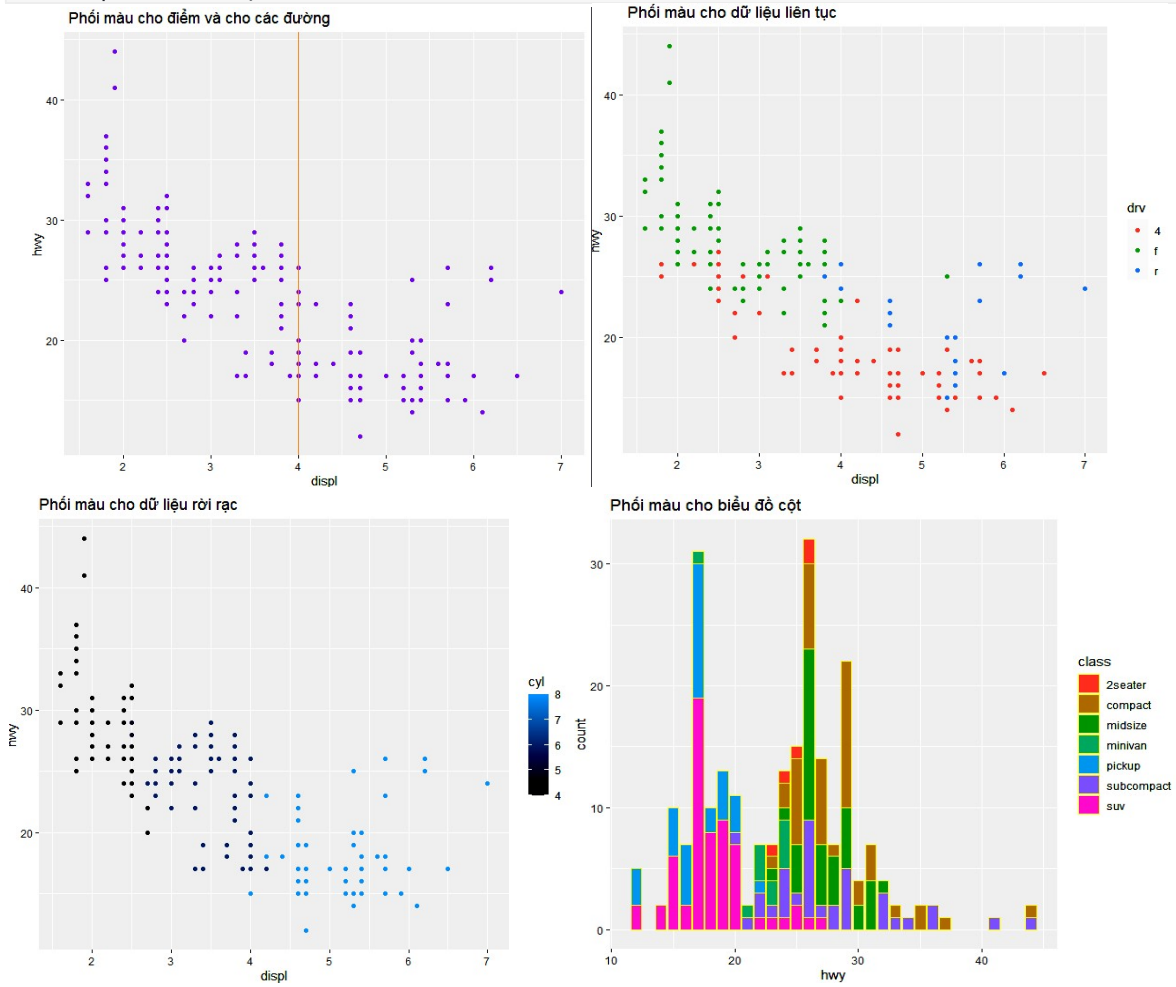
f) Phối màu sắc, tô màu, thang đo

- ***Phối màu***

Để phối màu sắc của *các đối tượng biểu đồ* (geoms/shapes) ví dụ như điểm, cột, đường, ô, v.v. chúng ta sẽ điều chỉnh color = (màu *bên ngoài*) hoặc fill = (màu *bên trong*), riêng đối với geom_point(), ta chỉ có thể điều khiển color =, để xác định màu của điểm. Khi thiết lập màu hoặc tô màu, chúng ta có thể sử dụng tên màu được R nhận dạng như "red" (xem danh sách các màu đầy đủ gõ ?colors trong cửa sổ soạn thảo hoặc ấn F1).

Ví dụ 1.11. Ví dụ dưới đây mô tả một vài cách phối màu kết hợp màu bên trong và bên ngoài.

```
# Phối màu cho điểm và cho các đường
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(color = "purple")+
  geom_vline(xintercept = 4, color = "orange")+
  labs(title = " Phối màu cho điểm và cho các đường ")
# Phối màu cho dữ liệu liên tục
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(mapping = aes(color = drv))+
  labs(title = " Phối màu cho dữ liệu liên tục ")
# Phối màu cho dữ liệu rời rạc
ggplot(data = mpg, mapping = aes(x = displ, y = hwy))+
  geom_point(mapping = aes(color =cyl))+
  labs(title = "Phối màu cho dữ liệu rời rạc")
# Phối màu cho biểu đồ cột
ggplot(data = mpg, mapping = aes(x = hwy))+
  geom_bar(mapping = aes(fill = class), color = "yellow")+
  labs(title = "Phối màu cho biểu đồ cột")
# kết quả hiển thị:
```



Hình 1.10: Một số cách phối màu.

- ***Thang đo cho yếu tố trang trí (thẩm mỹ)***

Khi gán một biến với một yếu tố thẩm mỹ của biểu đồ (vd: x =, y =, fill =, color =…), biểu đồ sẽ hiển thị một thang đo/chú giải, trên đó có thể là các giá trị liên tục, rời rạc, ngày tháng, v.v. tùy thuộc vào kiểu dữ liệu của biến được chỉ định. Nếu ta có nhiều yếu tố thẩm mỹ được gán tới biến, biểu đồ sẽ có nhiều thang đo.

Chúng ta có thể kiểm soát các thang đo bằng hàm scales_() thích hợp. Các hàm scale của ggplot() có 3 phần được viết như sau: scale_aesthetic_method().

- Phần đầu tiên, scale_(), là cố định.
- Phần thứ hai, aesthetic, là tên yếu tố thẩm mỹ bạn muốn điều chỉnh thang đo (_fill_, _shape_, _color_, _size_, _alpha_…). Các tùy chọn ở đây cũng bao gồm _x_ và _y_.
- Phần thứ ba, method, sẽ là một trong số các tùy chọn sau _discrete(), continuous(), _date(), _gradient(), hoặc _manual(), tùy thuộc vào kiểu dữ liệu củabiến và cách chúng ta muốn kiểm soát nó. Có những tùy chọn khác, tuy nhiên những lựa chọn trên thường được sử dụng nhất.

- ***Các đối số của hàm Scale***

Mỗi loại thang đo có những đối số riêng của chúng, mặc dù cũng có những sự trùng nhau (Truy vấn hàm chẳng hạn như ?scale_color_discrete trong cửa sổ R console để xem tài liệu về các đối số của hàm).

Với thang đo liên tục, sử dụng breaks = để cung cấp một chuỗi giá trị tới seq() (đặt to =, from =, và by =). Thiết lập expand = c(0,0) để loại bỏ không gian đệm xung quanh các trục (điều này có thể được sử dụng trên bất kỳ thang đo của trục _x_ hoặc _y_).

Với thang đo rời rạc, ta có thể điều chỉnh thứ tự của các giá trị với breaks =, và cách các giá trị hiển thị với đối số labels =, cung cấp một vectơ ký tự cho mỗi cái đó. Chúng ta cũng có thể loại bỏ NA dễ dàng bằng cách đặt na.translate = FALSE.

- ***Điều chỉnh thủ công***

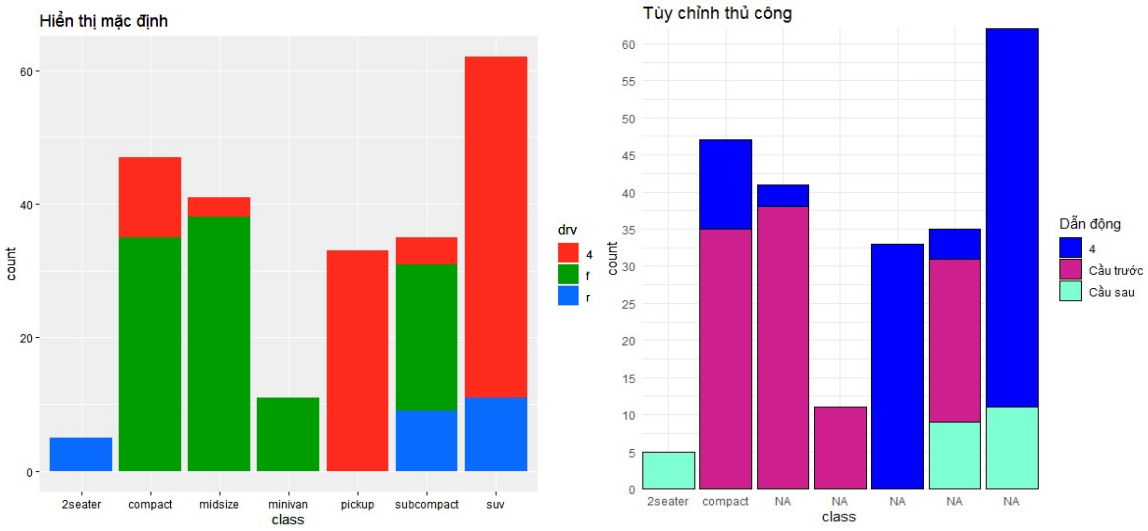
Chúng ta có thể sử dụng các hàm scaling "một cách thủ công" để gán màu sắc như mong muốn.

- Gán màu cho các giá trị dữ liệu với đối số values =.
- Cụ thể màu sắc cho giá trị NA với na.value =.
- Thay đổi cách các giá trị được viết trong chú giải với đối số labels =.
- Thay đổi tiêu đề chú giải bằng name =.

Ví dụ 1.12. Dưới đây, chúng ta tạo một biểu đồ cột và hiển thị cách nó xuất hiện theo mặc định, sau đó với ba thang đo được điều chỉnh - thang đo trục y, thang đo trục x, và điều chỉnh thủ công cách tô màu (màu bên trong cột).

```
# Hiển thị mặc định
ggplot(data = mpg)+
  geom_bar(mapping = aes(x = class, fill = drv))+
```

```
  labs(title = "Hiển thị mặc định")
# Hiện thị tùy chỉnh
ggplot(data = mpg)+
    geom_bar(mapping = aes(x = class, fill = drv), color = "black")+
    theme_minimal()+
    scale_y_continuous(expand = c(0,0),breaks = seq(from = 0,to = 70,by = 5))+
    scale_x_discrete(  expand = c(0,0),drop = FALSE, na.translate = FALSE,
    labels = c("2seater", "compact"))+
    scale_fill_manual(
    values = c("f" = "violetred","r" = "aquamarine","4" = "blue"),
    labels = c("f" = "Cầu trước", "r" = "Cầu sau","4 hướng"),name = "Dẫn động",
    na.value = "grey")+
  labs(title = "Tùy chỉnh thủ công")
# kết quả hiện thị:
```



Hình 1.11: Điều chỉnh thang đo ghi chú.

- ***Thang đo trên các trục***

Khi dữ liệu được ánh xạ tới các trục của biểu đồ, chúng cũng có thể được điều chỉnh bằng các lệnh scales. Phổ biến là điều chỉnh hiển thị của một trục (ví dụ: trục y) được ánh xạ tới một biến có dữ liệu liên tục.

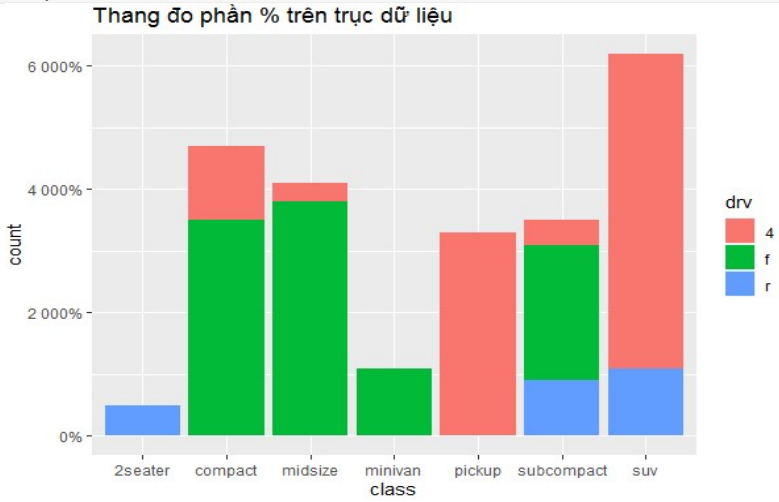
Chúng ta có thể điều chỉnh độ chia hoặc hiển thị của giá trị trong ggplot bằng cách sử dụng scale_y_continuous(). Như đã lưu ý ở trên, sử dụng đối số breaks = để cung cấp một chuỗi các giá trị sẽ đóng vai trò là "ngắt các khoảng giá trị" dọc theo thang đo. Đây là những giá trị mà các số sẽ hiển thị. Đối với đối số này, ta có thể cung cấp một vectơ c() chứa các giá trị để chia thang đo theo mong muốn hoặc bạn có thể cung cấp một chuỗi số thông thường bằng cách sử dụng hàm seq() từ base R. Hàm seq() này chấp nhận to =, from =, và by =. (xem code trong ví dụ 1.113).

- ***Hiển thị phần trăm trên trục***

Nếu giá trị dữ liệu ban đầu là tỷ lệ, chúng ta có thể hiển thị chúng dưới dạng phần trăm với "%" bằng cách cung cấp labels = scales::percent trong lệnh scales command. Ngoài ra, có một giải pháp thay thế là chuyển đổi các giá trị thành ký tự và thêm ký tự "%" vào cuối, cách tiếp cận này sẽ gây ra phức tạp vì dữ liệu sẽ không còn là các giá trị số liên tục.

Ví dụ 1.13. Thêm thang đo phần % vào cột dữ liệu.

```
# Hiển thị mặc định
ggplot(data = mpg)+
  geom_bar(mapping = aes(x = class, fill = drv))+
  labs(title = "Thang đo phần % trên trục dữ liệu")+
scale_y_continuous(labels = scales::percent )
# Kết quả hiển thị:
```



Hình 1.12: Thang đo hiển thị phần trăm.

- ***Thang đo log***

Một số dữ liệu khi hiển thị trên biểu đồ có khoảng cách (metric) khá lớn, dẫn tới khó quan sát hoặc dữ liệu biểu diễn vượt ra ngoài khung hình của biểu đồ. Khi đó việc biến đổi một trục liên tục sang thang đo log sẽ khắc phục được những hạn chế này. Cách chuyển rất đơn giản bằng cách thêm trans = "log2" vào lệnh scale.

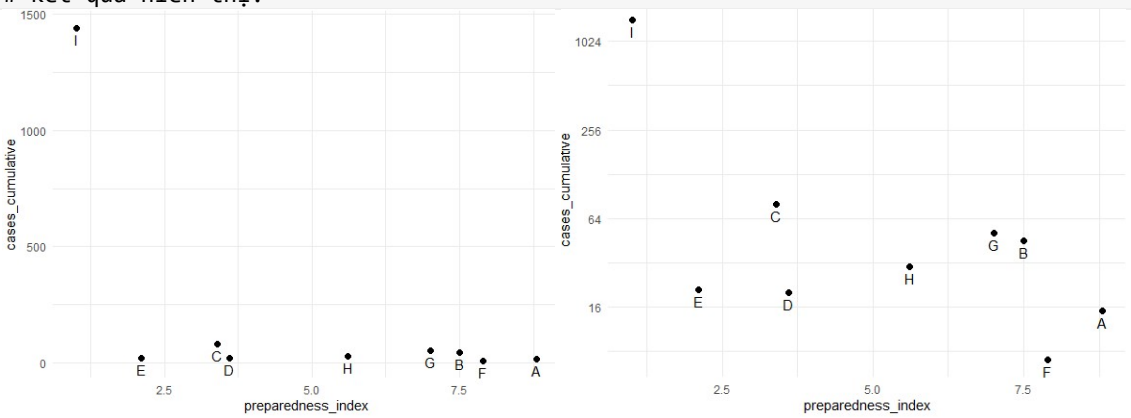
Ví dụ 1.14. Chúng ta tạo một data frame với các biến region; preparedness_index và cases_cumulative như sau:

```
plot_data <- data.frame(
  region = c("A", "B", "C", "D", "E", "F", "G", "H", "I"),
  preparedness_index = c(8.8, 7.5, 3.4, 3.6, 2.1, 7.9, 7.0, 5.6, 1.0),
  cases_cumulative = c(15, 45, 80, 20, 21, 7, 51, 30, 1442))
# Kết quả hiển thị:
##   region preparedness_index cases_cumulative
## 1      A               8.8              15
## 2      B               7.5              45
## 3      C               3.4              80
## 4      D               3.6              20
```

```
## 5      E               2.1              21
## 6      F               7.9               7
## 7      G               7.0              51
## 8      H               5.6              30
## 9      I               1.0            1442
```

Ta thấy giá trị tích lũy ở vùng "I" lớn hơn đáng kể so với tất cả các vùng khác. Trong những trường hợp như thế này, ta chọn hiển thị trục y bằng thang đo log để có thể thấy sự khác biệt giữa các vùng với ít trường hợp tích lũy hơn.

```
# Giá trị trục y thực tế
preparedness_plot <- ggplot(data = plot_data,
       mapping = aes(x = preparedness_index, y = cases_cumulative))+
  geom_point(size = 2)+
  geom_text(    mapping = aes(label = region), vjust = 1.5)+
  theme_minimal()
preparedness_plot
# Giá trị trục y biến đổi theo thang đo log
preparedness_plot+
  scale_y_continuous(trans = "log2")
# kết quả hiển thị:
```



Hình 1.13: Dữ liệu qua phép biến đổi thang đo log.

- ***Thang đo Gradient***

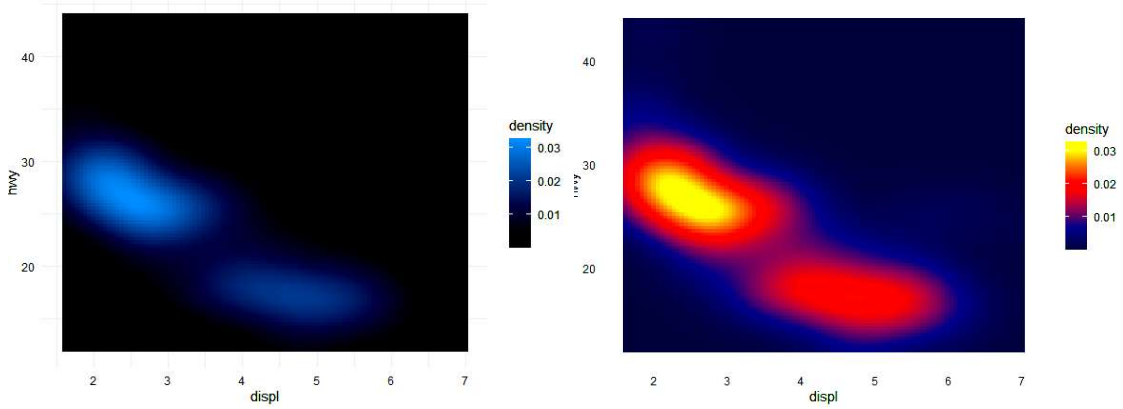
Tô màu theo thang đo gradient liên quan đến bản nhiệt. Các giá trị mặc định thường khá dễ chịu, nhưng ta có thể muốn điều chỉnh các giá trị, điểm cắt, v.v.

Chúng ta xét ví dụ sau đây với biểu đồ nhiệt, lần lượt điều chỉnh thang đo màu gradien.

Ví dụ 1.15. Xét biểu đồ nhiệt với màu gốc và màu hiệu ứng "plasma"

```
trans_matrix <- ggplot( data = mpg,mapping = aes(x = displ, y = hwy))+
  stat_density2d(
     geom = "raster",mapping = aes(fill = after_stat(density)),contour = FALSE)+
  theme_minimal()
trans_matrix
```

```
trans_matrix + scale_fill_viridis_c(option = "plasma")
# kết quả hiển thị:
```

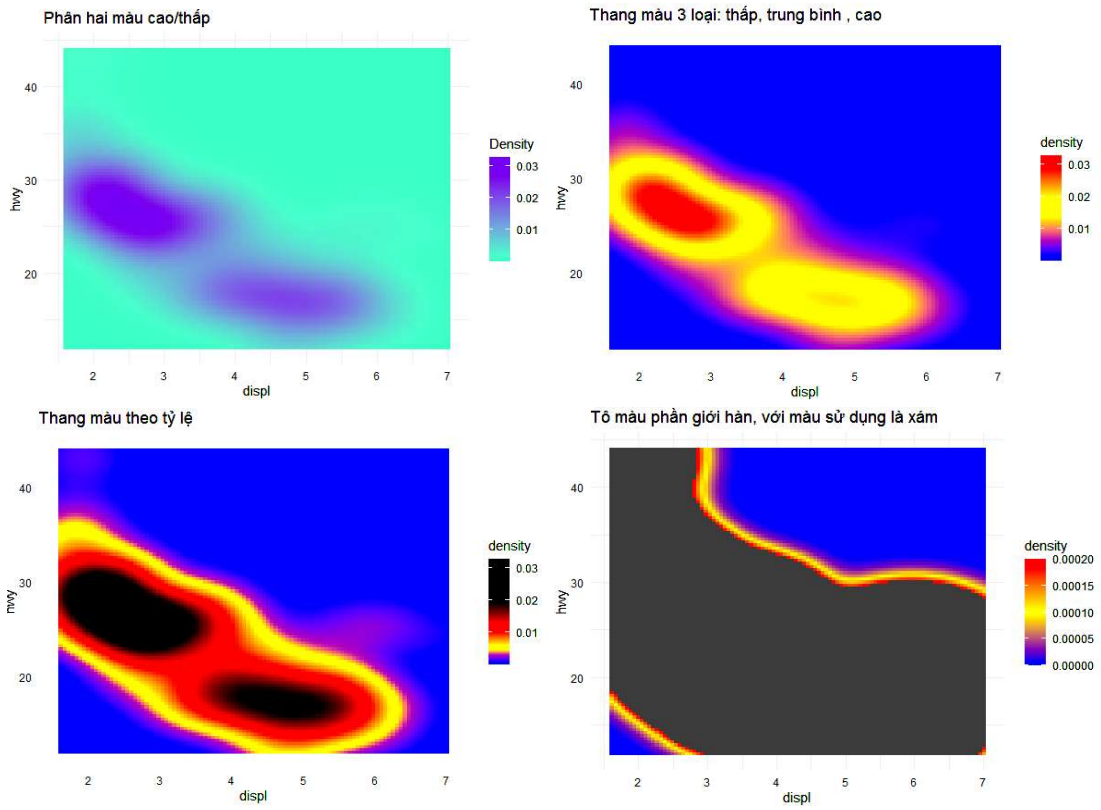


Hình 1.14: Biểu đồ dữ liệu với hiệu ứng gốc và "plasma".

Tiếp theo, chúng ta sẽ so sánh kết quả khi điều chỉnh các điểm ngắt của thang đo qua một số hàm sau:

- scale_fill_gradient() nhận hai màu (cao/thấp).
- scale_fill_gradientn() nhận một vectơ có độ dài màu bất kỳ tới values = (các giá trị trung gian sẽ được nội suy).
- Sử dụng scales::rescale() để điều chỉnh cách định vị màu sắc dọc theo gradient; nó sẽ cân chỉnh lại vectơ vị trí nằm giữa 0 và 1.

```
trans_matrix +
  scale_fill_gradient(low ="aquamarine",high ="purple",
                      na.value ="grey", name = "Density")+
  labs(title = "Phân hai màu cao/thấp")
# Thang màu 3 loại
trans_matrix +
  scale_fill_gradientn( colors = c("blue", "yellow","red") )+
  labs(title = "Thang màu 3 loại: thấp, trung bình , cao")
# Điều chỉnh màu theo tỷ lệ
trans_matrix +
  scale_fill_gradientn(
    colors = c("blue", "yellow","red", "black"),
    values = scales::rescale(c(0, 0.05, 0.07, 0.10, 0.15, 0.20, 0.3, 0.5)))+
  labs(title = "Thang màu theo tỷ lệ")
# Sử dụng các giá trị giới hạn để tô màu
trans_matrix +
  scale_fill_gradientn(
    colors = c("blue", "yellow","red"), limits = c(0, 0.0002))+
  labs(title = "Tô màu phần giới hàn, với màu sử dụng là xám ")
# kết quả hiện thị
```



Hình 1.15: Một số tùy chỉnh thang màu qua các hàm gradian.

g) Lưu trữ, chỉnh sửa và xuất bản biểu đồ

- ***Lưu biểu đồ***

Mặc định khi chạy lệnh ggplot(), biểu đồ sẽ được in ở cửa số Plots của RStudio. Tuy nhiên, bạn cũng có thể lưu biểu đồ dưới dạng một đối tượng bằng cách sử dụng toán tử gán <- và đặt tên cho nó. Biểu đồ sẽ không được in ra trừ khi ta gọi tên của đối tượng. Ta cũng có thể in nó bằng cách đưa tên biểu đồ vào hàm print(), nhưng điều này chỉ cần thiết trong một số trường hợp nhất định chẳng hạn như khi biểu đồ được tạo bên trong một *vòng lặp for* để in nhiều biểu đồ cùng một lúc.

Ví dụ 1.16. lệnh lưu biểu đồ hình 1.1 vào biến có tên Bieu_do1

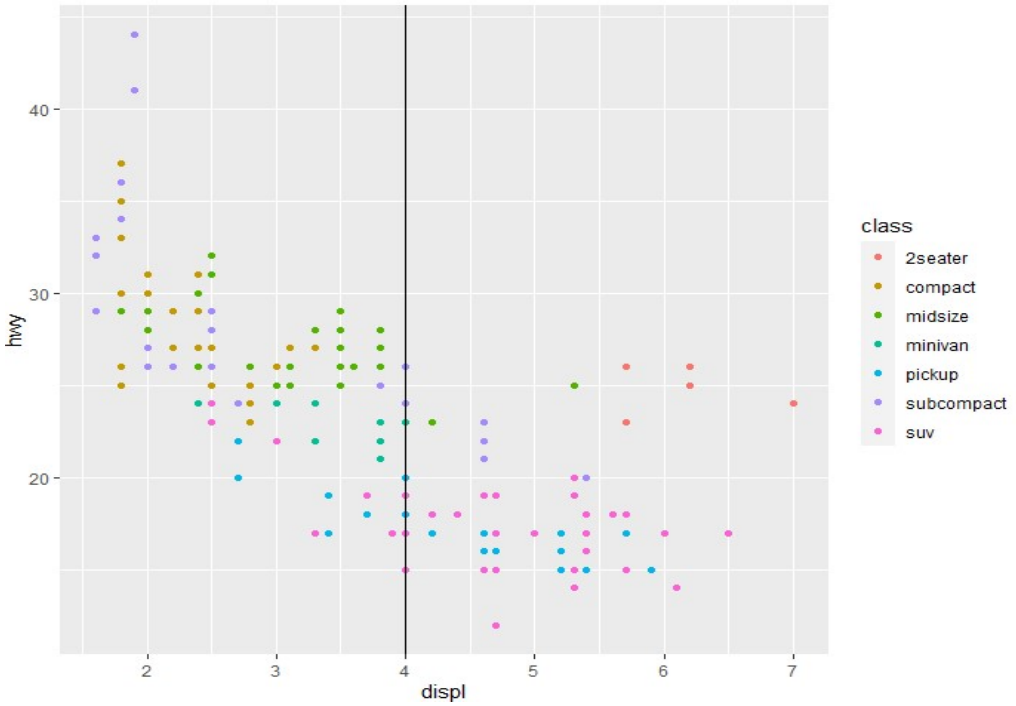
```
Bieu_do1<-ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy,color=class))
# in biểu đồ
Bieu_do1
```

- ***Chỉnh sửa biểu đồ đã lưu***

Một điểm hay của **ggplot2** là ta có thể gán tên cho một biểu đồ (như bên trên), và sau đó thêm các lớp mới hoặc chỉnh sửa bắt đầu bằng tên của nó. Chúng ta không cần phải lặp lại tất cả các lệnh đã tạo ra biểu đồ ban đầu.

Ví dụ 1.17. Ta chỉnh sửa đổi biểu đồ Bieu_do1 đã được lưu ở Ví dụ 1.16 ở bên trên, thêm một trục dọc tại dung tích động cơ bằng 4, chúng ta chỉ cần thêm dấu + và bắt đầu thêm các lớp bổ sung vào biểu đồ.

```
Bieu_do1+
  geom_vline(xintercept = 4)
# kết quả hiển thị:
```



Hình 1.16: Chỉnh sửa biểu đồ đã lưu trữ.

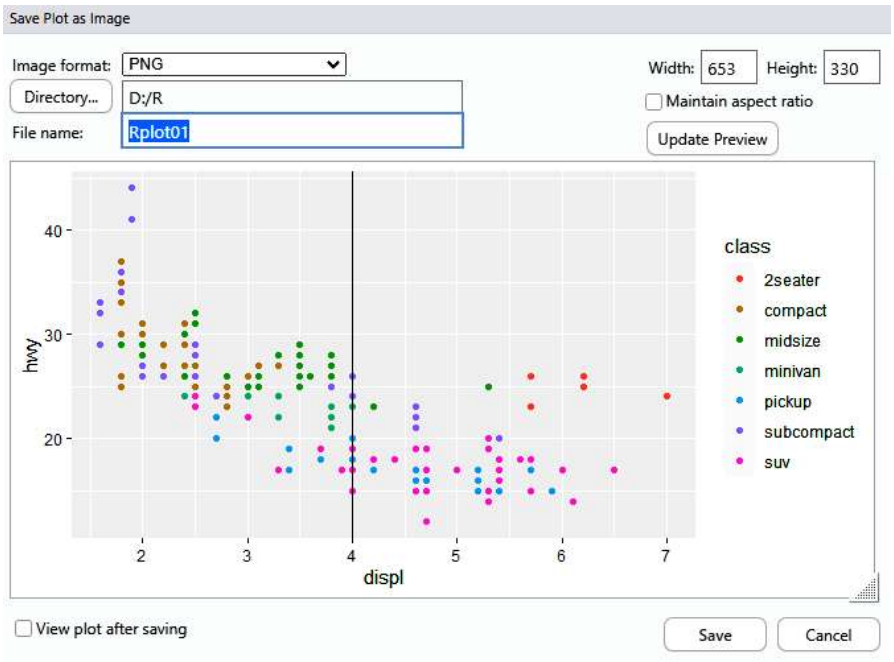
- ***Xuất bản biểu đồ***

Việc xuất bản biểu đồ được thực hiện dễ dàng với hàm ggsave() của package *ggplot2* hoặc chức năng Export trong Rstudio.

- Với hàm ggsave(), có thể được tiến hành theo hai cách:
  - Chỉ định tên của đối tượng biểu đồ, sau đó là đường dẫn tệp và tên có phần mở rộng. Ví dụ: ggsave(Bieu_do1, here("plots", " Bieu_do1.png")).
  - Chạy lệnh chỉ với một đường dẫn tệp, để lưu biểu đồ gần nhất được in ra.Ví dụ: ggsave(here("plots", " Bieu_do1.png")).

Chúng ta có thể xuất dưới dạng tệp png, pdf, jpeg, tiff, bmp, svg, hoặc một số định dạng khác, bằng cách chỉ định phần mở rộng tệp trong đường dẫn tệp. Hơn nữa, ta cũng có thể chỉ định các đối số width =, height =, và units = (“in”, “cm”, hoặc “mm”), và chỉ định dpi = để điều chỉnh độ phân giải của biểu đồ (vd: dpi = 300). Xem hướng dẫn chi tiết về hàm bằng cách gõ ?ggsave trong Rstudio.

- Với Export trong Rstudio, chúng ta có thể lựa chọn save image; pdf hoặc copy to clipboard,… Khi chọn save image chúng ta sẽ có 1 bảng thông số như hình dưới đây:



Hình 1.17: Xuất bản biểu đồ qua lệnh Import trong Rstudio.

Chúng ta điền các đối số width =, height =,…. phù hợp với mục đích sử dụng.