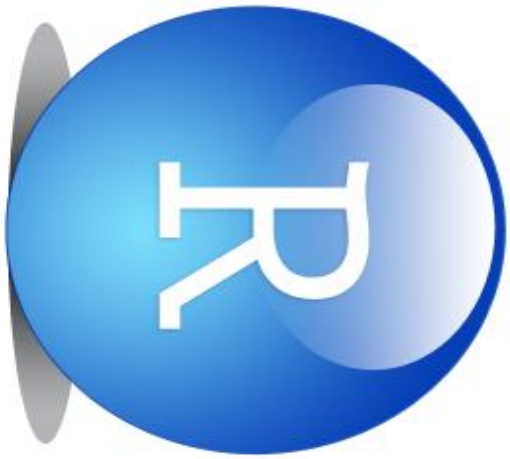
LẬP TRÌNH R CƠ BẢN

TẠ QUANG TÙNG
(KIBO)
Toán Tin

MỤC LỤC



Bài 1: Chương trình đầu tiên

a) **Cách in ra màn hình:**
- print(“Hello World”)
- cat(“string”)
- “Hello Rstudio”
- Các phép tính toán: 1 + 2 + 3

b) **Kết hợp giữa các biến khác kiểu dữ liệu:**
- print(paste(value, “String”))
- formattedString <- sprintf("y = %s + %s * x", a, b)

c) **Cách ghi chú, chú thích theo dòng:**
- Sử dụng dấu # để ghi chú:
  Vd: # Đây là chú thích

d) **Tải gói lệnh sử dụng:**
- library(“<the package’s name>”)
  Vd: > library(“RMySQL”)
- ?<nameFunction> : kiểm tra xem cách sử dụng hàm

Bài 2: Các phép toán cơ bản

a) Các phép toán tử số học:

Phép toán	Ví dụ
Phép cộng	123 + 234
Phép trừ	3234 - 532
Phép nhân	123 * 456
Phép chia	100 / 25
Lũy thừa	2 ^ 3
Chia lấy phần nguyên	8 %/% 3
Chia lấy phần dư	8 %% 3

b) Toán tử Logic:

- Toán tử và: & hoặc &&
- Toán tử hoặc: | hoặc ||
- Phép phủ định: !
- Ngoại trừ (&& và ||) → các toán tử khác đều có tính "VECTO HÓA"

=> giúp tính toán 2 vecto mà không cần dùng vòng lặp

c) Khởi tạo biến và kiểu dữ liệu:

- Cách gán biến:
  - Val_x <- "value"
  - Val_x = "value"

- Các kiểu dữ liệu:

Character: dạng chuỗi ký tự	fruit <- "Apple"
Integer: dạng số nguyên	integer_variable <- 186L
Numberic: dạng số thực	weight <- 63.5 height <- 182
Complex: dạng số phức	complex_value <- 4 + 2i
Logical: gồm 2 giá trị TRUE, FALSE	bool <- TRUE

Bài 3: Cấu trúc dữ liệu trong R

1) **Vector:**
   - Có kiểu dữ liệu như mảng → lưu trữ dữ liệu (mảng 1 chiều)
   - Hàm: c()
   - Gán biến cho vecto:
     - X = c(value1, value2, …)
     - XnameVal <- c(a:b)
   - Cách truy xuất phần tử trong vecto
     - X[i] : lấy giá trị truy xuất từ phần tử thứ i
     - X[-i] : bỏ đi phần tử thứ i
     - X[c(TRUE, FALSE, …)] : nếu TRUE thì in ra, FALSE thì không lấy

   VD: x <- c(1,2,3,4,5) hoặc x <- c(1:5)

2) **List:**
   - Danh sách list lưu trữ nhiều kiểu dữ liệu khác nhau (kiểu Tuple)
   - Hàm: list()
   - Gán biến cho list:

```
listL <- list(
    key1 = c(val1, val2, …),
    key2 = c(val1, val2, …),
    …
)
```

   VD:

```
listA <- list(
    a = c(1, 2, 3),
    b = c("Apple", "Orange"),
    c = c(TRUE, FALSE)
)
```

   - Truy xuất lấy phần tử trong 1 list, ta sử dụng dấu $: listName$key
     VD: listA$a # Lấy biến a trong list

3) **Factor:**
   - Là vecto lưu trữ biến định tính
   - Hàm: factor()
   - Công thức:
     - nameFactor <- c(string1, string2, …)
     - Factoring <- factor(nameFactor)

     VD:

```
sex_vec <- c("female", "female", "female", "male", "female", "male")
sex_fac <- factor(sex_vec)
```

     => Hiển thị tần suất xuất hiện (“female”, “male”)

- Truy cập các phần tử như vector

4) <u>Matrix:</u>

- Là mảng 2 chiều để lưu trữ dữ liệu cùng kiểu với nhau (mảng 2 chiều)
- Hàm: matrix()
- Công thức:
  - Matrixing = matrix(data = c(val1, val2, …), nrow = i)
  - Matrixing <- matrix(c(val1, val2, …), ncol = j)

  VD: matrix1 <- matrix(data=c(-3,2,893,0.17),ncol=2)

- Truy xuất dữ liệu cần 2 chỉ số (hàng, cột)
  - matrix[a, b] # lấy phần tử ở hàng a, cột b
  - matrix[a, -b] # lấy phần tử ở hàng a, bỏ cột b
  - matrix[a,] # lấy các phần tử ở hàng a

- Đếm số phần tử thỏa mãn điều kiện:
  - lst = matrix(...)
  - d = lst[lst <điều kiện>] # Hiển thị những số thỏa mãn
  - length(d) == sum(lst <điều kiện>) # Đếm số phần tử thỏa mãn
  - sum(dat$Column <điều kiện>)

Bài 4: Câu lệnh điều kiện và vòng lặp

a) Câu lệnh điều kiện IF – ELSE IF - ELSE:

```
if (<điều kiện>) {
        # Thực hiện câu lệnh này nếu đúng
} else if {<điều kiện>) {
        # Thực hiện câu lệnh này nếu đúng
} else {
       # Thực hiện câu lệnh này nếu đúng

}
```

- Phải có dấu ngoặc nhọn { } khi sử dụng

b) Hàm IFELSE rút gọn:

ifelse(test, YES, NO)

- test là biểu thức điều kiện
- YES là giá trị trả về nếu đúng
- NO là giá trị tra về nếu sai

- Thay thế cho cấu trúc if – else khi điều kiện đơn giản
- Hàm có tính vector hóa

c) Vòng lặp FOR và WHILE:

Vòng lặp For	Vòng lặp While

for (i in <vector>) {
  # Thực hiện các câu lệnh
}
 	
return kq
	while (<điều kiện>) {
	  # Thực hiện các câu lệnh
	}

d) Họ hàm lặp APPLY:

+ Ngoài sử dụng vòng lặp, ta sử dụng các hàm apply(), lapply(), sapply()

- Họ hàm apply():

apply(X, MARGIN, FUN)

- X là 1 vecto hoặc matrix
- MARGIN = 1 đối với hàng, MARGIN = 2 đối với cột
- FUN là tên hàm được định nghĩa

- Họ hàm lapply():

lapply(X, FUN)

- X là vecto hoặc matrix
- FUN là hàm có sẵn hoặc tự định nghĩa

=> Luôn trả về kết quả là 1 list

- Họ hàm sapply():

sapply(X, FUN)

- X là vecto hoặc matrix
- FUN là hàm có sẵn hoặc tự định nghĩa
  => Hàm trả về vecto hoặc matrix

Bài 5: Hàm Function

1. **Cấu trúc hàm function:**

```
<function_name>(arg1, arg2 =
<default>, ...)
```

- <function_name> là tên của hàm
- arg1 (bắt buộc), arg2, … là tham số của hàm
- Nếu không truyền tham số đầu vào từ arg2 trở đi thì hàm sẽ sử dụng mặc định <default> để tính toán

2. **Những cách khởi tạo hàm khác nhau:**

```
# Nếu không cần lưu kết quả của
hàm
        <function_name>(<args>)
# Nếu cần lưu kết quả của hàm
        result <-
<function_name>(<args>)
```

- Cách truyền tham số <args>:
  - Truyền tham số theo tên:

```
# Tham số có tên <arg> sẽ nhận giá trị <value>
        <function_name>(<arg> = <value>, ...)
```

  - Truyền tham số theo vị trí:

```
# Tham số thứ nhất trong hàm sẽ nhận <value1>, tham số thứ hai nhận
<value2>
        <function_name>(<value1>, <value2>, ...)
```

  - Vừa truyền cả hai loại tham số:

```
# Tham số thứ nhất trong hàm nhận <value1>, tham số tên <argument_n> nhận
<value2>
        <function_name>(<value1>, <argument_n> = <value2>)
```

=> đặt tên biến theo kiểu snake_case (biến dài phải có ý nghĩa)

3. **Công thức và cấu trúc hàm:**

+ Công thức:

```
Tenham <- function(<các tham số>) {
        # Nội dung của hàm
        return (ketqua)
}
```

VD: Tạo hàm tính tổng từ 1 đến x

```
# Cách khởi tạo hàm function
tong = function(x){
  sum  = 0
  for(i in 1:length(x)) {
    sum  = sum + x[i]
  }
  return(sum)
}
tong(c(1,2,3))
```

Bài 6: Một số hàm toán học cơ bản trong R

1. **Một số hàm toán học cơ bản:**
   - abs(x): lấy trị tuyệt đối của x
   - sqrt(x): lấy căn bậc hai của x
   - sin(x): lấy hàm lượng giác sin()
   - cos(x): lấy hàm lượng giác cos()
   - tan(x): lấy hàm lượng giác tan()
   - log(x): lấy hàm logarit cơ số e là ln == loge()
   - log10(x): lấy hàm logarit cơ số 10
   - exp(x): lấy hàm e^x
   - round(x): hàm làm tròn số
   - toupper(x): in hoa chuỗi

   => Các hàm toán học trên cũng có tính vecto hóa

2. **Một số hàm tính toán ma trận:**
   - Nhân ma trận, dùng toán tử %*%
   - Chuyển vị ma trận, dùng hàm t()
   - Tính định thức ma trận, dùng hàm det()
   - Tính nghịch đảo ma trận, dùng hàm solve()
   - Dùng * nhân từng phần tử ở vị trí tương ứng của A và B
   - Ma trận đơn vị trên đường chéo chính: diag(A)

3. **Một số hàm tính toán thống kê mô tả:**
   - Tính tổng: sum()
   - Trung bình: mean()
   - Trung vị: median()
   - Phương sai: var()
   - Độ lệch chuẩn: sd()
   - GTNN và GTLN: min() và max()
   - Thông tin thống kê một số biến số: summary()

4. **Một số hàm phân bố cơ bản:**
   - Mô phỏng mẫu dữ liệu thông qua phân bố cho trước
     - Phân bố chuẩn: rnorm()
     - Phân bố nhị thức: rbinom()
     - Phân bố Poisson: rpois()
     - Phân bố đều: runif()

Phân bố	Mô phỏng
Chuẩn	rnorm(n,mean,sd)
Nhị phân	rbinom(k,n,prob)
Poisson	rpois(n, lambda)
Uniform	runif(x, min, max)
Nhị thức âm	rnbinom(n,n,prob)
Beta	rbeta(n,shape1,shape2)
Khi bình phương	rchisq(x,df)
Hình học	rgeom(n,prob)
Mũ	rexp(n,rate)
Student	rt(n,df)
Fisher	rf(n,df1,df2)

- Các dạng bài tập phân phối chuẩn:
  - X là ĐLNN có phân phối chuẩn với kỳ vọng là EX và độ lệch tiêu chuẩn là SD:
    - P(X <= N) → pnorm(N, EX, SD)
    - P(X > N) = 1 – P(X <= N) → 1 - pnorm(N, EX, SD)
    - P(A < X < B) = P(X <= B) - P(X <= A)
    - Xác định a để P(X > a) = p → P(X <= a) = 1 – p => qnorm(1 - p, EX, SD)
    - Sinh ngẫu nhiên N giá trị → rnorm(N, EX, SD)
    - Hàm mật độ f(x) → dnorm <- dnorm(x, mean, sd)

5. **Phép toán đếm:**
   - Phép hoán vị: prod(n:k) → n(n-1)….(k+1)k
   - Phép tổ hợp: choose(n,k) → nCk

6. **Một số hàm tiện ích khác:**
   - Tính tổng và tích các phần tử trong vector: sum() và prod()
   - Kiểm tra thông tin hàm cần dùng: ?<name_function>
   - Tính số phần tử của vecto: length()
   - Sắp xếp các phần tử trong vector: sort(gt, decreasing = FALSE/TRUE)
   - Tạo một dãy số lặp: rep(a:b, times = <số lần lặp>)
   - Tạo một dãy có bước nhảy: seq(start, end, step)
     - step có thể là length.out/length = số phần tử được chia khoảng

7. **Tạo hàm tính Mode:**

```
# Tính mode
getmode = function(x){
    uniqv = unique(x)
    uniqv[which.max(tabulate(match(x, uniqv)))]
}
getmode(classInterval)
```

8. **Hàm tính tổng theo Hàng và Cột:**
   - colSums(): Tính tổng theo cột
   - rowSums(): Tính tổng theo hàng
   - colMeans(): Tính trung bình theo cột
   - rowMeans(): Tính trung bình theo hàng

Bài 7: Khởi tạo dữ liệu bảng DataFrame

1. **Nhập xuất dữ liệu cơ bản:**
   - Khởi tạo vector:
     - data1 <- c(val1, val2, …)
     - data2 <- c(value1, value2, …)
   - Tạo thành dữ liệu bảng: (SQL)
     - Data <- data.frame(data1, data2)
     - Data
   - Có thể dùng c() hoặc rep()
   - Tách dữ liệu có thể dùng → attach(data)

2. **Xử lý thư viện có sẵn và thư mục Excel:**
   - Xử lý dữ liệu với Excel: (CSV - Comma deliminated):
     + Cách lấy đường dẫn:

```
data = read.csv("ĐƯỜNG DẪN//Tenfile.csv)
dt = read.csv("Link\\Đường dẫn//Tenfile.csv), header = TRUE)
```

- Chú thích: "\\" = "/"
- Hiển thị dữ liệu ra màn hình: head(data)

VD:
data = read.csv("C:\\Users\\Admin\\OneDrive\\Documents\\Nhập môn phân tích dữ liệu//DuLieu1.csv", header = TRUE)

- So sánh cách đọc file CSV và file XLSX:

```
+ Đọc file .csv bằng hàm: read.csv()
+ Đọc file .xlsx bằng hàm: read_excel() trong gói lệnh readxl
xlsx_example <- readxl_example("datasets.xlsx")
read_excel(xlsx_example)
```

3. **Thực hiện với file CSV có sẵn trên trên R:**
   - Tải thư viện: install.packages("NameLibrary")
   - Thư viện chứa Boston: library(MASS)
   - Đọc dữ liệu Boston: Boston hoặc data(Boston)
   - Số chiều của KGVT: dim(Boston)
   - Hiển thị dữ liệu bảng: View(Boston)

4. **Xử lý dữ liệu thêm cột hàng:**
   - Cách lấy dữ liệu cột: df$column

   - Xử lý dữ liệu với hàm Factor(): # Dựa vào giá trị cụ thể

```
df <- mtcars
df$columnNew <- factor(df$column1, levels = c(value1, value2), label = c("valueNew1", "valueNew2")
df$columnNew
```

⇨ Tạo thêm một cột “columnNew” chứa các giá trị valueNew1, valueNew2 dựa vào điều kiện trong column1

- Xử lý dữ liệu với hàm Cut(): # Dựa vào giá trị khoảng

```
df$columnNew <- cut(df$column1, breaks = c(-Inf, a, b, Inf), labels = c("key1", "key2", "key3"))
df$columnNew
```

⇨ Tạo thêm một cột “columnNew” chứa các giá trị key1, key2, key3 lần lượt trong các khoảng cho sẵn dựa vào column1

- Chuyển đổi hóa dữ liệu giữa các loại vector:
  + as.factor(): chuyển thành factor
  + as.character(): chuyển thành vector chữ
  + as.numeric(): chuyển thành vector số

5. Thao tác lọc dữ liệu:

- Lọc dữ liệu: (xử lý dataFrame dựa trên điều kiện ở trong ngoặc vuông [ ])
- Lấy dữ liệu ở cột: df$column
  VD:

```
df <- matcars
data1 <- df[a, ]      # Lấy dữ liệu ở hàng a
data2 <- df[ ,b]      # Lấy dữ liệu ở cột b
data3 <- df[c(row1, row2, row3), c(col4, col5, col6)]    # Lấy dữ liệu ở hàng 1,2,3 và cột 4,5,6
manual <- df[df$column <điều kiện>, <# nếu đúng sẽ thực hiện> (có thể không có)]
unique(manual)     # Lấy những dữ liệu sao cho không trùng nhau
```

6. Trích ghép dữ liệu:

- Sử dụng hàm Merge để ghép hai dữ liệu lại với nhau

```
df1 <- data.frame(row1 = c("a", "b", "c"), row2 = c("d", "e", "f"))
df2 <- data.frame(col1 = c("a", "b", "c"), col2 = c("Hello", "Geeks", "gfg"))

# Sử dụng hàm Merge để ghép hai dữ liệu với nhau
df <- merge(df1, df2, by.x ="row1", by.y ="col1")
print(df)
```

7. Tạo bảng tần số:

- Công thức: table(dataDL)
- Tạo ra bảng tần số số lần xuất hiện của dữ liệu

Bài 8: Một số hàm xử lý dữ liệu khác

1. **Hàm biến ngẫu nhiên Sample():**
   - Công thức:

     sample(<khoảng random>, size = 'kích cỡ hàng', replace = T)

     VD:

```
VD1: id = sample(1:1000, size = 2345, replace = T)
VD2: DTX = sample(seq(1,10,0.25), 2345, replace = T)
```

2. **Hàm in tất cả dữ liệu lớn bị ẩn:**

```
# Hiển thị tất cả các giá trị
options(max.print = <maxiumNumber>)
```

3. **Hàm thay thế dữ liệu Replace():**
   - Khởi tạo dữ liệu mới được thay thế dựa trên một số dữ liệu cũ và có chức năng như câu lệnh IF - ELSE duyệt từng giá trị bảng
   - Công thức:

```
NewColumn = dat$columnSOSANH
NewColumn = replace(NewColumn, điều kiện <dat$columnSOSANH>, "Thực thi nếu đúng")
dat = data.frame(dat, NewColumn)      # Gán dữ liệu NewColumn thành cột mới của bảng dat
```

   VD:

```
GPA = dat$DTB
GPA = replace(GPA, dat$DTB >= 9 & dat$DTB <= 10, "A+")
GPA = replace(GPA, dat$DTB >= 8.5 & dat$DTB < 9, "A")
GPA = replace(GPA, dat$DTB >= 8 & dat$DTB < 8.5, "B+")
GPA = replace(GPA, dat$DTB >= 7 & dat$DTB < 8, "B")
GPA = replace(GPA, dat$DTB < 7 & dat$DTB >= 6.5, "C+")
GPA = replace(GPA, dat$DTB < 6.5 & dat$DTB >= 5.5, "C")
GPA = replace(GPA, dat$DTB < 5.5 & dat$DTB >= 5.0, "D+")
GPA = replace(GPA, dat$DTB < 5 & dat$DTB >= 4.0, "D")
GPA = replace(GPA, dat$DTB < 4 & dat$DTB >= 0, "F")

dat = data.frame(dat, GPA)
```

4. **Hàm truy tìm vị trí dữ liệu Which():**
   - Tìm vị trí dữ liệu thỏa mãn yêu cầu của điều kiện
   - Công thức:

     which(<điều kiện>)

     VD:

```
miniNuts <- min(data$AllRent, na.rm = T)
which(data$AllRent == miniNuts)
```

5. **Lập bảng dữ liệu có dựa vào điều kiện thỏa mãn:**

+ Công thức:

NewTable <- subset(dat, điều kiện <dat$Column>)

VD: dating <- subset(dat, dat$City == "Boston") *# Lập bảng dữ liệu dựa trên điều kiện "Boston"*

Bài 9: Xử lý dữ liệu bị thiếu

1. **Lỗi dữ liệu dataframe (NA):**
   + Có một số thông số dữ liệu bị tróng (NA)
   + Cách xử lý:
     - Xóa bỏ các hàng, các cột bị thiếu dữ liệu
     - Điền dữ liệu các ô bị thiếu bằng giá trị trung bình, trung vị, …
     - Xây dựng mô hình để điền dữ liệu bị thiếu

2. **Các cách thực thi dữ liệu bị thiếu:**
   a. ***Xóa bỏ dữ liệu hàng thiếu:***
   + Công thức: na.omit(<dữ liệu data>)
   VD:

```
data <- read.csv("melb_data.csv")
dim(data)          # dim = a
data2 <- na.omit(data)
dim(data2)         # dim = a - b
```

   b. ***Thay thế dữ liệu định lượng bị thiếu:***
   + Thống kê dữ liệu bị thiếu trong từng cột bằng is.na() và colSum()
   + Công thức: colSum(is.na(<data>))
   VD:

```
data <- read.csv("melb_data.csv")
dim(data)
colSums(is.na(data))
```

   c. ***Công thức thay thế dữ liệu bị thiếu:***
   + Thay thế dữ liệu bị trống bằng dữ liệu khác (GTTB)
   + Công thức:

```
colSums(is.na(data))                # Thống kê dữ liệu bị thiếu
data[is.na(data$ColumnThieu), ]$ColumnThieu <- mean(data$ColumnThieu, na.rm = TRUE)
```

   VD:

```
data <- read.csv("melb_data.csv")          # Đọc file
dim(data)
colSums(is.na(data))              # Thống kê dữ liệu bị thiếu
data[is.na(data$BuildingArea), ]$BuildingArea <- mean(data$BuildingArea, na.rm = TRUE)
colSums(is.na(data))          # Thống kê dữ liệu bị thiếu sau khi thay thế
```

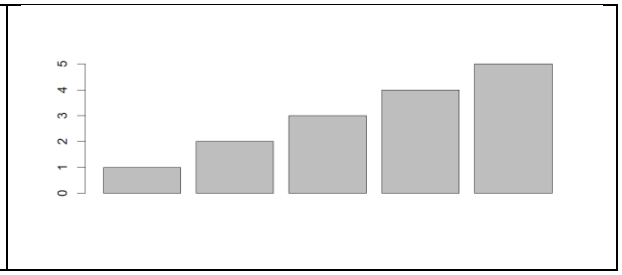
Bài 10: Các dạng biểu đồ

1. **Biểu đồ cột Barplot():**

   + Công thức: **barplot(height)** với height là vector chứa chiều cao của các cột

   VD minh họa:

```
height <- c(1, 2, 3, 4, 5)
barplot(height)          # Biểu đồ cột
```



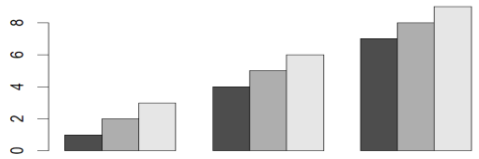
2. **Biểu đồ cột chồng, cột kép:**

   + Công thức: **barplot(height, beside = TRUE/FALSE, names.arg = tên cột kép)**

   - height là một ma trận, truyền thêm tham số beside = TRUE nếu vẽ biểu đồ cột kép

   VD minh họa:

```
matrix <- matrix(1:9, nrow = 3)
barplot(matrix, beside = TRUE)
```

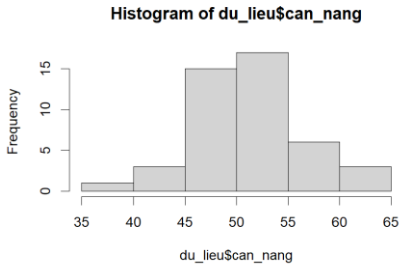


3. **Tổ chức đồ Histogram():**

   + Công thức: **hist(data)** với data là vector dữ liệu

   VD minh họa:

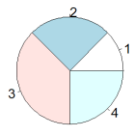
```
hist(du_lieu$can_nang)
```



4. **Biểu đồ tròn Pie():**

   + Công thức: **pie(area, labels = ghi_chú)** với area là tỷ lệ diện tích giữa các hình quạt

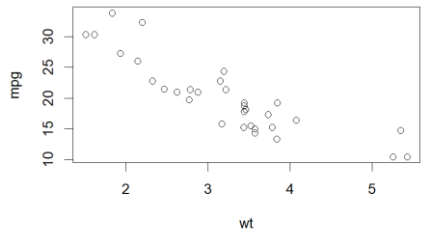
VD minh họa:

area <- c(1,2,3,2) pie(area)	

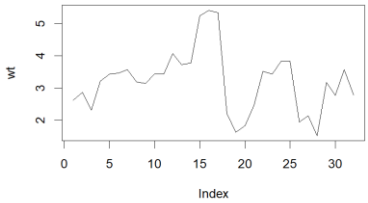
5. Biểu đồ tán xa Plot():

+ Công thức: **plot(data1, data2)** với data1, data2 là 2 vector dữ liệu muốn vẽ

VD minh họa 1:

attach(mtcars) plot(wt, mpg)	

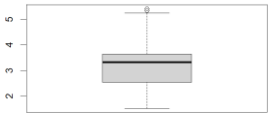
VD minh họa 2:

plot(wt, type = "l")	

6. Biểu đồ hộp Boxplot():

+ Công thức: **boxplot(data)** với data là dữ liệu mà ta muốn vẽ biểu đồ hộp

VD:

attach(mtcars) boxplot(wt)	

- Hiển thị nhiều đồ thị trên khung:
  par(mfrow = c(a,b)) # Chia khung vẽ đồ thị thành a hàng, b cột
  dev.off()

Bài 11: Ghi chú, chú thích cho biểu đồ

1. **Chú thích biểu đồ cột:**

+ Để đặt tên và thêm chú thích cho biểu đồ, ta thêm các tham số sau vào hàm vẽ biểu đồ:

- main: Tiêu đề chính của biểu đồ
- xlab: Chú thích cho trục hoành
- ylab: Chú thích cho trục tung
- sub: Tiêu đề phụ của biểu đồ

- xlim: đặt giới hạn trục hoành
- ylim: đặt giới hạn trục tung
- col: Thêm màu cho biểu đồ

(dùng hàm colors(_) để biết các màu có thể dùng trong biểu đồ)

VD:

```
barplot(
        c(vector),
        main="Tiêu đề chính",
        xlab="Tiêu đề cho trục hoành",
        ylab="Tiêu đề cho trục tung",
        sub="Tiêu đề phụ"
)
```

```
barplot(
        c(vector),
        xlim = c(x, y),
        ylim = c(z, t),
        col = c("red", "blue", "green")
)
```

2. **Thêm chú thích cho biểu đổ chung:**

+ Công thức: ***legend(vitri, legend, fill, col)***

- vitri: cặp (x, y) ứng với tọa độ muốn đặt chú thích hoặc đặt chú thích bằng các từ khóa sau đây:
  - "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"
- legend: là vector chứa nội dung của các chú thích
- fill hoặc col: là vector chứa màu của các chú thích
  - col: để ký hiệu biểu đồ tán xạ, biểu đồ đường
  - fill: biểu đồ cột, biểu đồ tròn

VD:

```
legend(
        "topleft",
        legend=c("Cột 1", "Cột 2", "Cột 3"),
        fill=c("red", "blue", "green")
)
```

3. **Xử lý ảnh và sao lưu biểu đồ:**

- Sử dụng hàm pdf(), png(), jpeg() để mở 1 file, ta sẽ dùng file này để lưu biểu đồ
- Vẽ biểu đồ bình thường
- Dùng hàm ***dev.off()*** để đóng file

VD:

```
# Tạo hình ảnh, mở file bằng hàm pdf(), png(), jpeg()
jpeg("rplot.jpg", width = 350, height = 350)    # Lưu biểu đồ

# Vẽ biểu đồ và chỉnh sửa trong hình ảnh
plot(
  x = mtcars$wt, y = mtcars$mpg,
  pch = 16, frame = FALSE,
  xlab = "wt", ylab = "mpg", col = "#2E9FDF"
  )
dev.off()      # Đóng file
```

4. Loại bỏ Outlier và vẽ lại biểu đồ:

- Công thức: quantile(vector, xác suất)

VD:

```
# Loại bỏ outlier khi vẽ biểu đồ
# Cách 1:
q_1 <- quantile(wt, 0.25)
q_3 <- quantile(wt, 0.75)
iqr <- q_3 - q_1
wt_no_out <- wt[wt >= q_1 - 1.5*iqr & wt <= q_3 + 1.5*iqr]
boxplot(wt_no_out)
# Cách 2:
outliers <- boxplot(wt, plot=FALSE)$out
wt_no_out <- wt[!wt %in% outliers]
boxplot(wt_no_out)
```