Chương 12

GIỚI THIỆU NGÔN NGỮ R

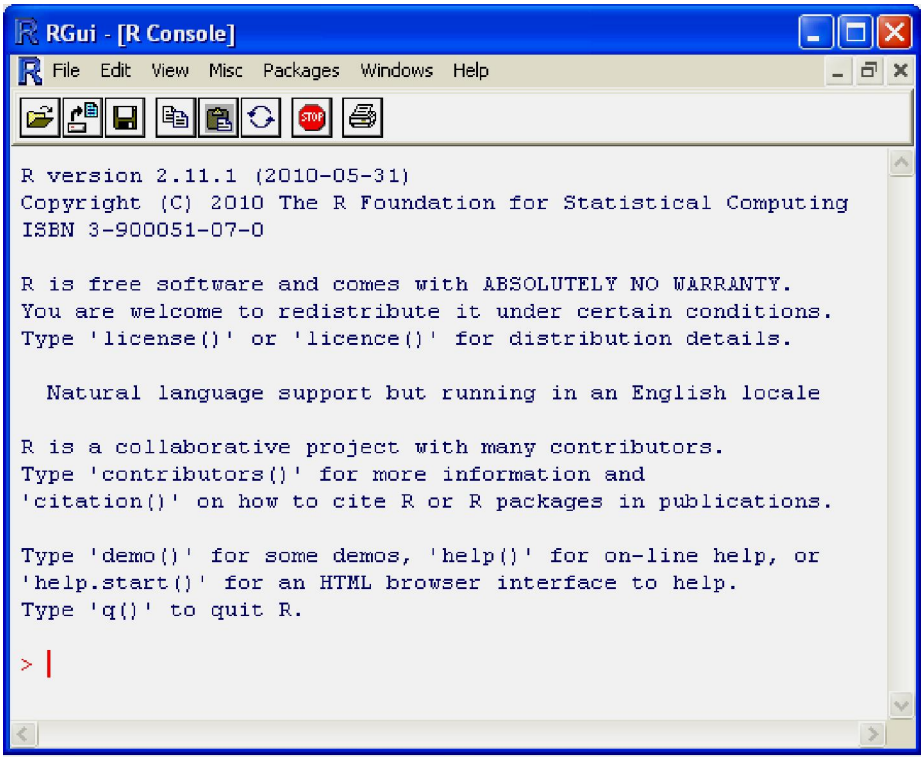
R là một ngôn ngữ lập trình hàm cấp cao vừa là một môi trường dành cho tính toán thống kê. **R** hỗ trợ rất nhiều công cụ cho phân tích dữ liệu, khám phá tri thức và khai mỏ dữ liệu nhưng lại là phần mềm miễn phí mã nguồn mở. Hơn nữa **R** rất dễ học và có thể phát triển nhanh các ứng dụng tính toán xác suất thống kê, phân tích dữ liệu (tham khảo tài liệu [4, 5, 7, 9, 10]).

12.1 R CĂN BẢN

Ngôn ngữ **R** được đề xuất bởi R. Ihaka và R. Gentleman [5] là phần mềm miễn phí mã nguồn mở chạy trên nhiều nền phần cứng như Intel, PowerPC, Alpha, Sparc và nhiều hệ điều hành khác nhau như Unix, Linux, Windows, Mac. **R** có thể thay thế cho ngôn ngữ **S+** (phần mềm thương mại) trong các tính toán thống kê ứng dụng [1, 2]. Ngôn ngữ **S+** có nguồn gốc từ ngôn ngữ **S** [8] được phát triển bởi ở phòng nghiên cứu AT&T Bell.

R rất dễ học và có thể phát triển nhanh các ứng dụng khai mỏ dữ liệu trong thời gian ngắn nhờ nhiều công cụ tích hợp sẵn dùng như khả năng lập trình, kiểu dữ liệu phong phú, các hàm thống kê, giải thuật học tự động và các giao diện truy vấn dữ liệu, hiển thị dữ liệu.

Như trình bày ở phần trên, chúng ta cũng có ngôn ngữ **S+** có nguồn gốc từ ngôn ngữ **S** được phát triển bởi ở phòng nghiên cứu AT&T Bell dùng cho phân tích dữ liệu, tính toán thống kê, mô phỏng, hiển thị. Tuy có chức năng tương tự như **R** nhưng **S+** là phần mềm thương mại. Môi trường **R** có độ ổn định rất cao. **S+** có môi trường đồ họa để lập trình. **R** hiện cũng hỗ trợ môi trường phát triển **R GUI** và nhiều trình soạn thảo khác như Eclipse, Kate. Tài liệu hỗ trợ cho người dùng và người phát triển đều phong phú như nhau. **R** tính toán có phần nhanh hơn về tốc độ so với **S+**. Các thông tin liên quan đến **R** có thể tham khảo tại địa chỉ http://www.r-project.org/ và ngôn ngữ **S+** tại địa chỉ http://www.in-sightful.com/.



Hình 12.1:** Môi trường lập trình ngôn ngữ **R

Trước hết, chúng ta có thể tải về một phiên bản của **R** và các gói thư viện từ địa chỉ của trang web http://cran.r-project.org. Tiến hành cài đặt, và gọi thực thi môi trường **R** như hình 12.1. Để thêm các gói thư viện vào môi trường lập trình **R**, chúng ta sử dụng các chức năng được cung cấp từ menu Packages.

R là ngôn ngữ lập trình hàm cấp cao. Tất cả các công việc được làm thông qua hàm. Ta có thể truyền các thông số cho hàm. Giá trị trả về của hàm có thể được lưu vào một đối tượng biến nhờ vào phép gán = hoặc <-, ví dụ như ta ghi giá trị trung bình của vectơ **x** vào biến **mx**, sử dụng lệnh: **mx = mean(x)** hoặc **mx <- mean(x).**

Giá trị trả về của hàm được lưu vào biến mặc định tên **.Last.value**, lệnh **print()** hay **cat()** dùng để hiển thị đối tượng.

R cung cấp trợ giúp trực tuyến thông qua hàm **help()** hay **?tên_hàm**. Lệnh **help.start()** cũng cung cấp trợ giúp với giao diện Web.

R hỗ trợ nhiều kiểu dữ liệu như: luận lý (**true**, **false**), số nguyên, số thực, số phức, ký tự, chuỗi ký tự, vectơ, danh sách, ma trận, khung dữ liệu (data frame) và các phép toán như: +, -, *, /, **%%** (chia lấy phần dư), **%/%** (chia lấy phần nguyên), **!** (nghịch đảo), ^ (luỹ thừa), **%*%** (nhân ma trận), <, >, ==, >=, <=, **&** (và), **&&**, | (hoặc), ||, **<-** (gán), ->.

Một đối tượng khi tạo ra sẽ được lưu trong bộ nhớ cho đến khi ta thoát khỏi môi trường **R** bằng hàm **q()**, lúc này đối tượng có thể được lưu lại cho lần làm việc sau hoặc biến mất đi trong trường hợp người sử dụng không cần đến. Nhóm các hàm được bắt đầu bởi { và kết thúc bởi }.

Hàm **library()** sẽ liệt kê tất cả các thư viện sẵn dùng trong môi trường **R**, nếu muốn sử dụng thư việc nào chỉ cần gọi hàm **library(tên_thư_viện)**, ngoài ra **library(help=tên_thư_viện)** sẽ liệt kê các hàm sẵn có trong thư viện.

Để bắt đầu làm việc với **R**, từ dấu nhắc hệ thống ta gõ **R** chương trình sẽ vào môi trường làm việc ở chế độ gõ giao tiếp với dấu nhắc >. Nếu muốn nạp tập tin chương trình, ta có thể dùng hàm **source("tập_tin_chương_trình")**. Hàm **q()** dùng để thoát khỏi môi trường **R**.

Hàm **c()** có thể được dùng để tạo ra vectơ và gán trực tiếp đến đối tượng trong **R**, chẳng hạn như lệnh cho phép tạo vectơ **x** chứa 5 phần tử như sau:

```
x <- c(12, 54, 13, 10, 5)
```

Hàm **seq()** cho phép tạo ra dãy số trong **R**, ví dụ như lệnh sau cho phép tạo dãy số từ 6 đến 20 với khoảng cách là 0.5:

```
x <- seq(6, 20, by = 0.5)
```

Hàm **rep()** cho phép tạo ra véctơ trong **R** bằng cách lặp lại giá trị của phần tử, ví dụ như lệnh sau cho phép tạo dãy 10 số 1:

```
x <- rep(1, 10)
```

Các phép toán có thể được áp dụng trực tiếp lên kiểu véctơ. Ví dụ sau cho phép tính tổng bình phương của dãy số 1, 2, 3, 4, 5:

```
x <- c(1, 2, 3, 4, 5)
sq <- sum(x^2)
```

Nếu muốn đọc dữ liệu vectơ từ tập tin, ta có thể sử dụng hàm **scan**, ví dụ lệnh đọc dữ liệu từ tập tin **data_file** (giá trị các phần tử cách nhau bởi khoảng trắng) vào biến **x** như sau:

```
x <- scan("data_file")
```

Hàm **scan()** không tham số cho phép nhập dữ liệu từ bàn phím kết thúc bằng dòng trống. Ngoài ra, hàm **matrix** còn cho phép chuyển dữ liệu vectơ thành ma trận, ví dụ lệnh đọc dữ liệu từ tập tin **mat_file** (ma trận có 5 cột và giá trị các phần tử được lưu theo dòng) vào biến **mat** như sau:

```
mat <- matrix(scan("mat_file"), ncol = 5, byrow = T)
```

Để đọc một khung dữ liệu từ tập tin, ta có thể sử dụng hàm **read.table()**. **R** cũng hỗ trợ các hàm cho phép ghi lại các biến vào tập tin dữ liệu như **save()**, **write()**, **write.table()**.

Việc truy cập nội dung các phần tử của vectơ, ma trận, khung dữ liệu được thực hiện thông qua chỉ số được đặt trong cặp dấu **[]**. Để truy xuất đến toàn bộ cột hay dòng của một ma trận ta có thể đặc tả chỉ số của cột hay dòng đó, ví dụ như **x[,3]** để truy xuất đến cột thứ 3 của ma trận **x**.

Nếu muốn gán tên cho cột hay dòng của ma trận ta có thể sử dụng hàm **dimnames()**, ví dụ như lệnh sau cho phép gán tên **X** cho cột một, tên **Y** cho cột hai và tên **Z** cho cột ba của ma trận **mat**, vậy **mat[,2]** và **mat[,"Y"]** là như nhau.

```
dimnames(mat) <- list(NULL, c("X", "Y", "Z"))
```

Đối với khung dữ liệu, ta có thể dùng tên của biến để truy cập đến nội dung, chẳng hạn ta có khung dữ liệu tên **soil** có các biến **Ca**, **K** và **pH**, ta có thể truy cập nội dung biến **pH** với biểu thức **soil$pH**. *Chú ý rằng các định danh trong **R** đều phân biệt ký tự thường hoa*.

Ngoài ra, ta cũng có thể truy cập nội dung các biến của khung dữ liệu trực tiếp bằng tên biến nhờ vào hàm **attach**, ví dụ **attach(soil)**.

R cũng hỗ trợ các cho các tính toán thống kê như: **summary()**, **sample()**, **dnorm()**, **pnorm()**, **qnorm()**, **rnorm()**, **dunif()**, **punif()**, **qunif()**, **runif()**, **mean()**, **var()**, **sd()**, **cov()**, **cor()**, **lm()**, ...

Ví dụ sau đây minh họa cách sử dụng hàm **sample()** để lấy mẫu ngẫu nhiên có hoàn lại 10 số nguyên có giá trị từ 0 đến 9. Sau đó tính giá trị trung bình và độ lệch chuẩn của dãy số.

```
x <- sample(0:9, 10, replace = TRUE)
mx <- mean(x)
sx <- sd(x)
```

Nếu muốn sử dụng hàm mô phỏng phân phối chuẩn **rnorm()** để sinh dãy số ngẫu nhiên gồm 10 số với các tham số giá trị trung bình là 5 và độ lệch chuẩn bằng 1.5, ta có thể làm như sau:

```
x <- rnorm(10, mean = 5, sd = 1.5)
```

Ví dụ minh họa cách sử dụng hàm tính mật độ phân phối chuẩn **dnorm()** để ước tính xác suất của học sinh có điểm là 16.5, biết rằng điểm của học sinh tuân theo phân phối chuẩn với giá trị trung bình là 15, độ lệch chuẩn là 2.5, ta có thể sử dụng lệnh sau:

```
p <- dnorm(16.5, mean = 15, sd = 2.5)
```

Tiếp theo ví dụ trên để ước tính xác suất học sinh có điểm tối thiểu là 16.5, ta có thể sử dụng hàm tính xác suất chuẩn tích lũy **pnorm()** như sau:

```
p <- 1 - pnorm(16.5, mean = 15, sd = 2.5)
```

$$A = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Đối với các tính toán trên ma trận, ngoài phép toán nhân ma trận, **R** cung cấp các hàm như: **t()**, **det()**, **diag()**, **solve()**, **svd()**, **lower.tri()**, **upper.tri()**, **eigen()**.

Đầu tiên ta tạo ma trận **A[3x2]**, và ma trận **B** là ma trận chuyển vị của A có dạng như trên, trong **R** thực hiện công việc đó như sau:

```
A <- matrix(seq(1, 6, by = 1), nrow = 3)
B <- t(A)
```

Tạo ma trận đường chéo **I[3x3]** có giá trị 1:

```
I <- matrix(0, 3, 3)
diag(I) <- 1
```

Để nhân ma trận **A** và **B** lưu vào ma trận **C**, tiếp theo ta lấy định thức của C bằng hàm **det()** và giá trị riêng ma trận bằng hàm **eigen()**. Ta thực hiện lệnh sau:

```
C <- A%*%B
det(C)
eigen(C)
```

Giả sử ta có hệ phương trình tuyến tính sau đây:

$$\begin{cases} x_1 + x_2 = 3 \\ 3x_1 + x_2 = 7 \end{cases}$$

Để giải hệ phương ta có thể viết dạng ma trận **A.X = Y** như sau:

$$A = \begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{và} \quad Y = \begin{pmatrix} 3 \\ 7 \end{pmatrix}$$

Nghiệm phương trình $\mathbf{X = A^{-1}Y}$, trong **R** ta thực hiện như sau:

```
A <- matrix(c(1, 1, 3, 1), nrow = 2, byrow = TRUE)
Y <- matrix(c(3, 7), nrow = 2)
X <- solve(A)%*%Y
```

R cho phép đọc dữ liệu từ tập tin vào trong bộ nhớ để xử lý. Ngoài ra, **R** cũng cung cấp các gói thư viện tiện ích cho phép người sử dụng dễ dàng truy cập vào các cơ sở dữ liệu quan hệ khác như mySQL (**RMySQL**), PostgreSQL (**RPgSQL**), các cơ sở dữ liệu có thể truy cập qua trình điều khiển ODBC (**RODBC**).

Nếu chúng ta muốn truy cập vào dữ liệu chứa trong hệ quản trị cơ sở dữ liệu MySQL. Trước tiên ta cần:

- Nạp gói thư viện **RMySQL** bằng lệnh **library(RMySQL)**,
- Sau đó sử dụng hàm **dbConnect()** để nối kết đến cơ sở dữ liệu và
- Thực hiện các câu truy vấn với hàm **dbSendQuery()**,
- Lấy kết quả trả về với hàm **fetch()**, **dbGetInfo()**, **dbNextResult()**, **dbMoreResults()**,
- Cuối cùng kết thúc bằng việc giải phóng các kết quả sử dụng hàm **dbClearResult()** và đóng kết nối với cơ sở dữ liệu **dbDisconnect()**.

Bảng 12.1: Ví dụ minh họa truy xuất cơ sở dữ liệu từ MySQL

```
library(RMySQL)
drv <- dbDriver("MySQL")
conn <- dbConnect(drv, host="192.168.1.1", dbname="mydb", user="user",
password="1234")
rs <- dbSendQuery(conn, "select * from ring order by rand() limit 30")
data <- fetch(rs, n = -1)
dbClearResult(rs)
dbDisconnect(conn)
```

Ví dụ trên đây cho phép kết nối đến cơ sở dữ liệu tên **mydb** trên MySQL server địa chỉ **192.168.1.1**, số hiệu cổng dịch vụ là **3306** thông qua người dùng **user** và mật khẩu là **1234**, sau đó thực hiện câu truy vấn lấy ngẫu nhiên 30 mẫu tin từ bảng dữ liệu **ring** có 20 trường dữ liệu số, lưu vào ma trận **data**.

Trong trường hợp dữ liệu của chúng ta chỉ là dạng tập tin văn bản bình thường, chúng ta vẫn có thể xử lý được bằng cách sử dụng các hàm hỗ trợ của **R** (**read.table()**, **write.table()**, etc). Ví dụ như ta có tập dữ liệu lưu trong tập tin dạng văn bản có tên là **mydata.dat**, các dữ liệu được lưu theo dạng ma trận dòng cột, các giá trị được cách nhau bởi khoảng trắng. Chúng ta có thể đọc dữ liệu lên và lưu vào bảng dữ liệu tên là tab như sau:

```
tab <- read.table("mydata.dat", sep=" ")
tab <- as.matrix(tab)
```

Sau khi làm việc xong, chúng ta có thể ghi lại kết quả của bảng **res** vào tập tin **result.dat**, giá trị được cách nhau bởi khoảng trắng:

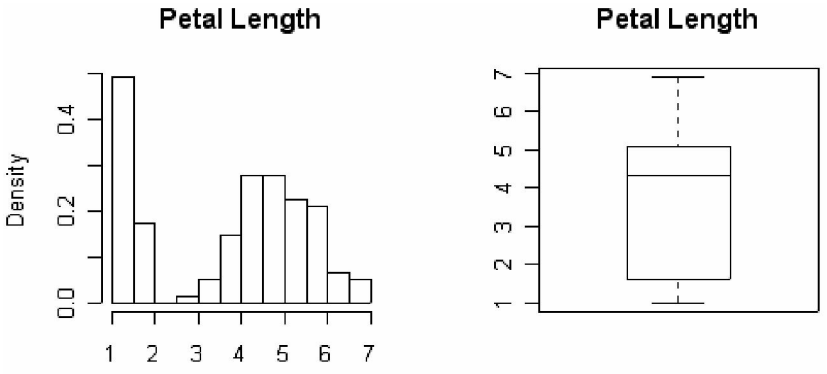
```
write.table(res, file="result.dat", sep=" ", col.names=FALSE, row.names=FALSE)
```

12.2 PHƯƠNG PHÁP HIỂN THỊ DỮ LIỆU

Trong phân tích thống kê, biểu đồ đóng vai trò rất quan trọng. Nếu biểu đồ được trình bày dễ hiểu, hợp lý, là phương tiện hiệu quả cung cấp cho nhà phân tích về thông tin quan trọng của dữ liệu cần phân tích. Phương pháp hiển thị dữ liệu thường dùng trong thống kê có thể kể đến là đồ thị hộp (box plot) và tổ chức đồ (histogram). Đồ thị hộp của 1 thuộc tính hiển thị giá trị nhỏ nhất, trung vị, lớn nhất, bách phân 25% và 75%. Tổ chức đồ hiển thị thông tin về phân bố dữ liệu của một thuộc tính.

Bảng 12.2: Ví dụ minh họa vẽ đồ thị hộp và tổ chức đồ

```
# doc du lieu, thuoc tinh thu 3 la Petal Length
data(iris)


# ve do thi to chuc do
hist(iris[,3], main = "Petal Length", ylab = "Density", prob = T)


# ve do thi hop
boxplot(iris[,3], main = "Petal Length")
```



Hình 12.2: *Đồ thị hộp và tổ chức đồ của thuộc tính* ***Petal Length*** *dữ liệu* ***iris***

Ví dụ trong hình 12.2 là đồ thị hộp và tổ chức đồ của thuộc tính **Petal Length** dữ liệu **iris**. Đoạn mã trong ngôn ngữ **R** cho phép thực hiện vẽ đồ thị hình 12.2 như bảng 12.2. Trong đó dòng lệnh 4 thực hiện vẽ tổ chức đồ. Dòng lệnh 6 vẽ đồ thị hộp.

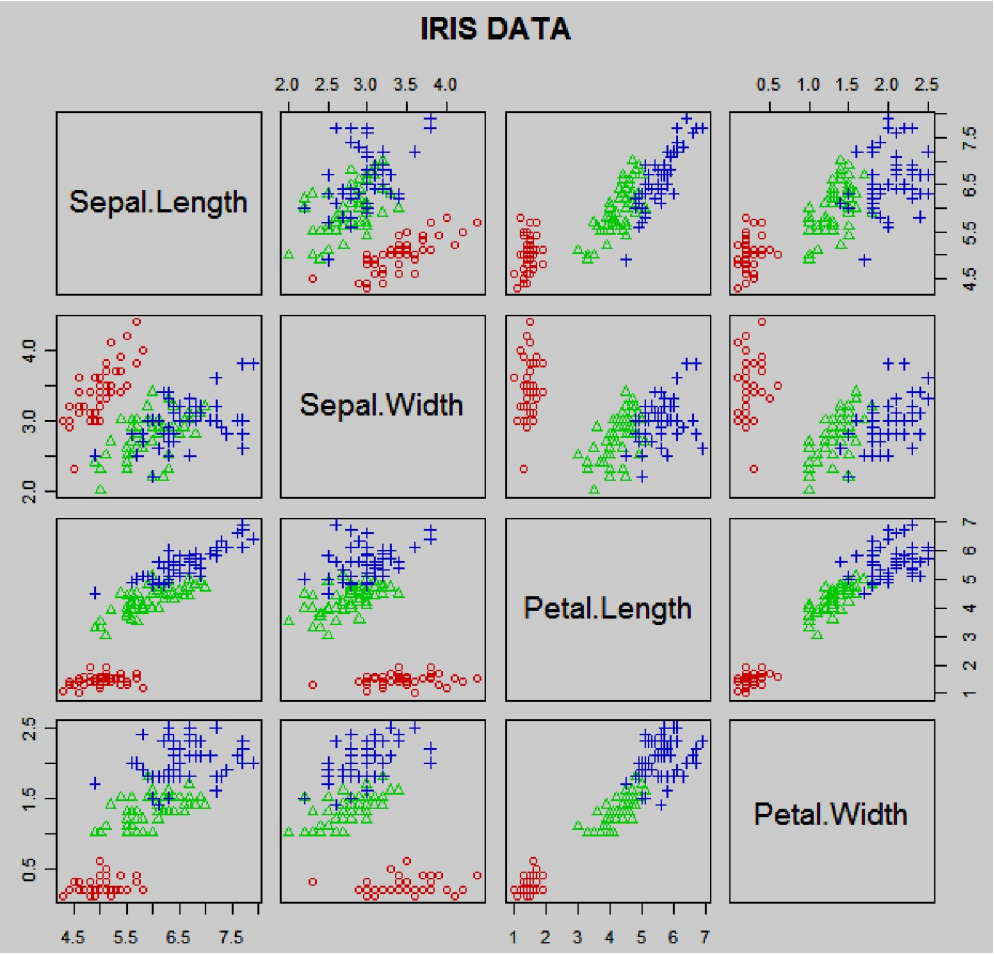
Cả hai đồ thị đều rất dễ hiểu, tuy nhiên chỉ làm việc được cho 1 thuộc tính, cung cấp các thông tin đơn giản. Để làm việc với dữ liệu nhiều chiều, chúng ta cần các phương pháp như scatterplot 2 chiều [3] và hệ trục tọa độ song song [6].

12.2.1 Phương pháp scatterplot 2 chiều trong R

Tập dữ liệu có ***n*** thuộc tính được hiển thị trong ma trận scatterplot 2 chiều, ở đó mỗi scatterplot 2 chiều trình bày dữ liệu của cặp thuộc tính A_i, A_j, mỗi phần tử được vẽ thành 1 điểm trong không gian 2 chiều của scatterplot.

Bảng 12.3: Ví dụ minh họa vẽ ma trận scatterplot 2 chiều

```
# doc du lieu
data(iris)
ncol <- length(iris[1,])

# ve ma tran scatterplot 2 chieu
pairs(iris[,-ncol], main = "IRIS DATA",  pch = c(1, 2, 3)[iris[,ncol]],
      col = c("red", "green", "blue")[iris[,ncol]])
```



Hình 12.3: *Ma trận scatterplot 2 chiều của dữ liệu* ***iris***

Ví dụ như ma trận scatterplot 2 chiều hiển thị tập dữ liệu **iris** (150 phần tử, 4 thuộc tính, 3 lớp tương ứng với màu của điểm dữ liệu) như hình 12.3. Để vẽ ma trận scatterplot 2 chiều cho ví dụ này, ta có thể viết đoạn mã trong **R** với hàm **pairs()** như bảng 12.3. Dòng lệnh 5 thực hiện vẽ ma trận scatterplot 2 chiều.

Phương pháp ma trận scatterplot 2 chiều dễ hiểu, hỗ trợ cho phát hiện mối tương quan của dữ liệu theo từng cặp thuộc tính, có thể phát hiện được nhóm dữ liệu, hiểu sơ lược về dữ liệu phức tạp hay không. Tuy nhiên khi số thuộc tính dữ liệu lớn lên, ma trận scatterplot có kích thước tăng nhanh (n^2 scatterplots với n là số thuộc tính dữ liệu). Phương pháp cũng chỉ hiển thị khoảng vài ngàn phần tử dữ liệu.

12.2.2 Phương pháp trục tọa độ song song trong R

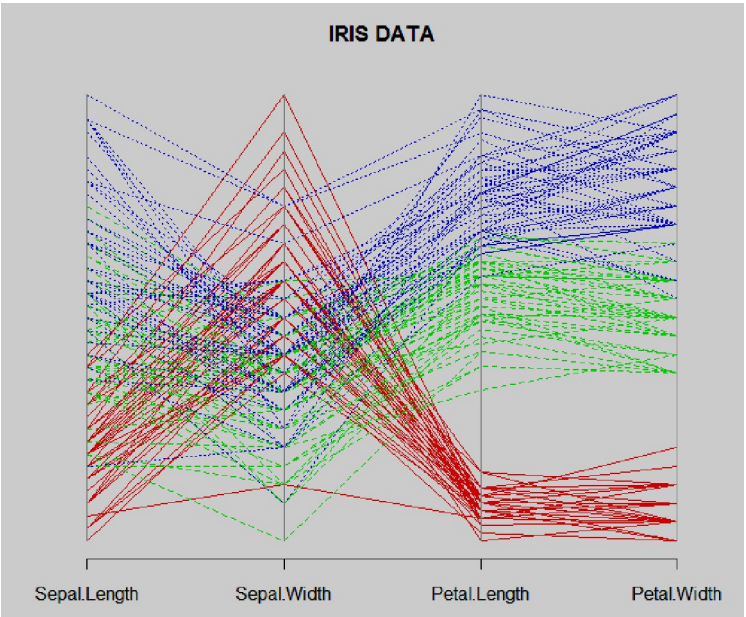
Inselberg [6] đề nghị phương pháp hiển thị dữ liệu với hệ trục tọa độ song song cho phép hiển thị cùng lúc rất nhiều thuộc tính (chiều). Dữ liệu có n thuộc tính trong hệ tọa độ Descartes được biểu diễn thành n trục tọa độ song song bằng nhau (một chiều trong hệ tọa độ Descartes là một trục tọa độ song song). Một phần tử trong hệ tọa độ Descartes trở thành 1 đường gấp khúc, giao điểm của đường gấp khúc với trục tọa độ song song chính là tọa độ chiều tương ứng

của điểm đó trong không gian Descartes. Phương pháp cho phép hiển thị cùng lúc khoảng 50 thuộc tính, vài ngàn phần tử dữ liệu. Hệ tọa độ song song dễ hiểu, hỗ trợ cho phát hiện nhóm dữ liệu, hiểu sơ lược về dữ liệu phức tạp hay không. Tuy nhiên thứ tự của các trục tọa độ ảnh hưởng lớn đến chất lượng hiển thị dữ liệu.

Bảng 12.4: Ví dụ minh họa hiển thị dữ liệu với hệ tọa độ song song

```
# nap thu vien MASS
library(MASS)

# doc du lieu
data(iris)
ncol <- length(iris[1,])

# hien thi du lieu voi he toa do song song
parcoord(iris[,-ncol], main = "IRIS DATA", lty = c(1, 2, 3)[iris[,ncol]],
         col = c("red", "green", "blue")[iris[,ncol]])
```

Trong gói thư viện **MASS** có hàm **parcoord()** cho phép thực hiện hiển thị dữ liệu với hệ trục tọa độ song song. Đoạn mã chương trình **R** trong bảng 12.4 cho phép hiển thị dữ liệu **iris** trong hệ tọa độ song song. Dòng lệnh 7 thực hiện hiển thị dữ liệu với hệ trục tọa độ song song. Kết quả hiển thị như hình 12.4 (màu của điểm dữ liệu tương ứng với lớp hay nhãn).



Hình 12.4: *Hiển thị dữ liệu* ***iris*** *với hệ tọa độ song song*

Tất cả các phương pháp đều có ưu, nhược điểm khác nhau. Chúng ta nên kết hợp các phương pháp hiển thị để tận dụng các ưu điểm và khắc phục được khiếm khuyết khi sử dụng đơn thuần một phương pháp. Cùng tập dữ liệu, mỗi phương pháp hiển thị cung cấp thông tin hữu dụng khác nhau.

12.2.3 Phương pháp hiển thị khác trong R

R còn cung cấp nhiều hàm tiện ích khác cho việc hiển thị dữ liệu. Thông thường **R** hỗ trợ hai loại hàm vẽ đồ họa:

- Mức cao (vẽ toàn bộ đồ thị bằng 1 lời gọi hàm duy nhất) bao gồm: **barplot()**, **boxplot()**, **contour()**, **coplot()**, **hist()**, **pairs()**, **persp()**, **plot()**, **pie()**,

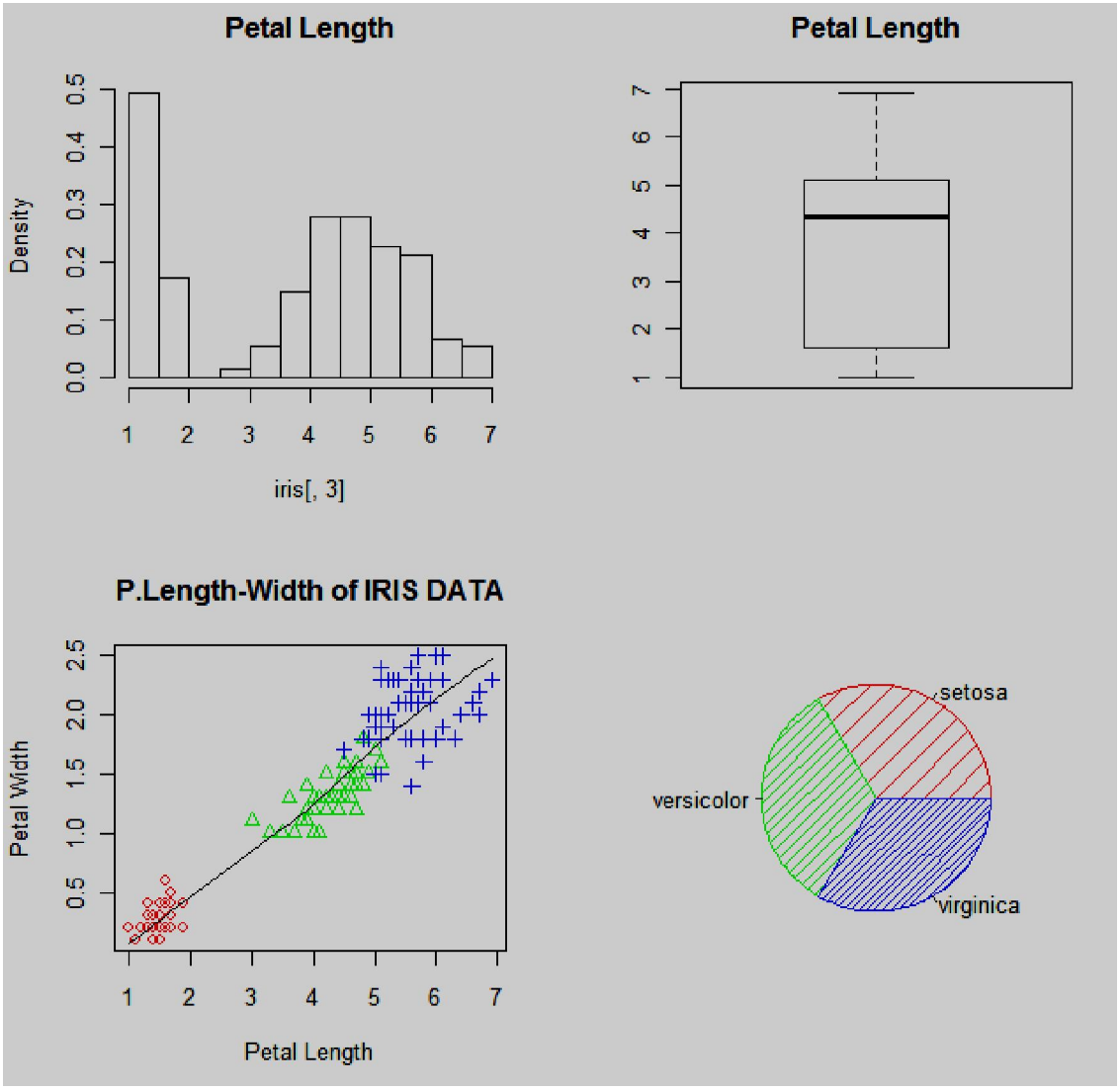
- Mức thấp (được sử dụng để thêm các thông tin vào đồ thị đã vẽ trước) bao gồm: **abline()**, **axis()**, **legend()**, **lines()**, **points()**, **polygon()**, **symbols()** và **text()**.

Ngoài ra ta còn có các hàm phụ trợ khác như tạm dừng với **par(ask=T)**, màn hình đồ họa có thể được chia thành dạng lưới cho phép trình bày nhiều đồ thị trên cùng màn hình thông qua hàm **par(mfrow=c(số_dòng, số_cột))**.

Ví dụ tiếp đây sẽ chia màn hình hiển thị thành 4 phần, phần 1 dùng để hiển thị tổ chức đồ cho thuộc tính Petal Length, phần 2 vẽ đồ thị hộp của thuộc tính Petal Length, phần 3 dùng để vẽ đồ thị của 2 thuộc tính Petal Length, Petal Width và đồ thị pie được vẽ trên phần còn lại như hình 12.5. Đoạn mã chương trình trong bảng 12.5 cho phép thực hiện công việc trên. Dòng lệnh 4 cho phép chia màn hình thành lưới 2x2.

Bảng 12.5: Ví dụ minh họa hiển thị dữ liệu iris với 4 phương pháp

```
# nap du lieu
data(iris)

# chia man hinh 2x2
op <- par(mfrow = c(2, 2))

# ve to chuc do thuoc tinh thu 3 la Petal Length
hist(iris[,3], main = "Petal Length", ylab = "Density", prob = T)

# ve do thi hop cua Petal Length
boxplot(iris[,3], main = "Petal Length")

# ve scatterplot 2 chieu cua p.length-width
plot(iris[,3:4], main = "P.Length-Width of IRIS DATA",
      xlab = "Petal Length", ylab = "Petal Width",
      pch = c(1, 2, 3)[iris[,ncol]],
      col = c("red", "green", "blue")[iris$Species])
# ve duong hoi quy
lines(lowess(iris[,3:4]))

#  ve do thi pie
nb1 <-  sum(c(1, 0, 0)[iris$Species])
nb2 <-  sum(c(0, 1, 0)[iris$Species])
nb3 <-  sum(c(0, 0, 1)[iris$Species])
pie(c(nb1, nb2, nb3), labels = c("setosa", "versicolor", "virginica"),
    density = c(10, 20, 30), col = c("red", "green", "blue"))
```



Hình 12.5: *Hiển thị dữ liệu* ***iris*** *với 4 phương pháp hiển thị*

12.3 HỒI QUY TUYẾN TÍNH

Hồi quy là phương pháp toán học được áp dụng thường xuyên trong thống kê để phân tích mối liên hệ giữa các hiện tượng kinh tế xã hội. Để minh họa cho vấn đề này, chúng ta xét một ví dụ rất đơn giản hồi quy tuyến tính đơn mà ở đó người ta cần nghiên cứu để biết chiều cao trung bình của trẻ dựa theo tháng tuổi. Bảng số liệu thu thập được trình bày trong bảng sau đây:

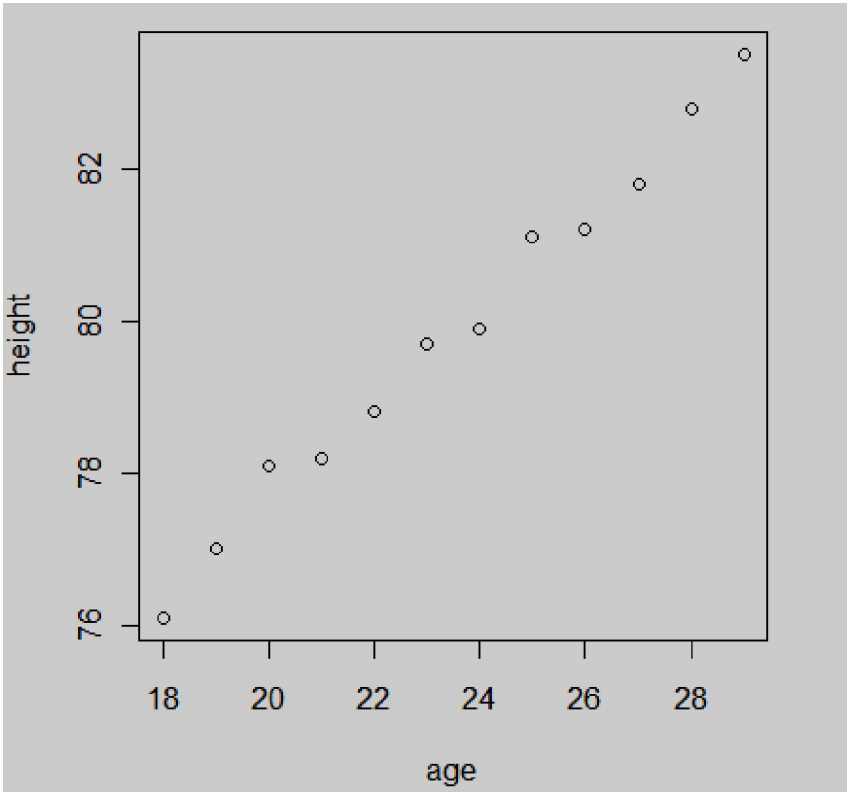
Bảng 12.6: Chiều cao trung bình của trẻ theo tháng tuổi

STT	Tháng tuổi	Chiều cao (cm)
1	18	76.1
2	19	77
3	20	78.1
4	21	78.2
5	22	78.8
6	23	79.7
7	24	79.9
8	25	81.1
9	26	81.2
10	27	81.8
11	28	82.8
12	29	83.5

Một cách trực quan, chúng ta có thể hiển thị bảng dữ liệu với scatterplot 2 chiều trong **R** như sau:

```
age=18:29
height=c(76.1,77,78.1,78.2,78.8,79.7,79.9,81.1,81.2,81.8,82.8,83.5)
plot(age,height)
```

Đồ thị scatterplot 2 chiều thu được đồ thị về chiều cao trung bình của trẻ theo tháng tuổi như hình 12.6.



Hình 12.6: *Đồ thị về chiều cao trung bình của trẻ theo tháng tuổi*

Nhìn vào đồ thị, chúng ta có thể thấy được mối liên quan giữa chiều cao và tuổi của trẻ có dạng tương tự như đường thẳng (tuyến tính). Mô hình hồi quy tuyến tính có dạng:

$y = \alpha + \beta x$

với α là chặn (intercept), β là độ dốc (slope).

Hai tham số α, β được ước tính từ bảng dữ liệu bằng phương pháp bình phương bé nhất (least squares):

$$\textbf{Min} \sum_{i=1}^{n} [y_i - (\alpha + \beta x_i)]^2$$

Trong ví dụ ở đây chiều cao height chính là y và độ tuổi age là x. Để thực hiện phân tích hồi quy tuyến tính cho vấn đề này, **R** cung cấp hàm **lm** cho phép tìm nhanh giá trị tham số α, β như sau:

```
res=lm(height ~ age)
```

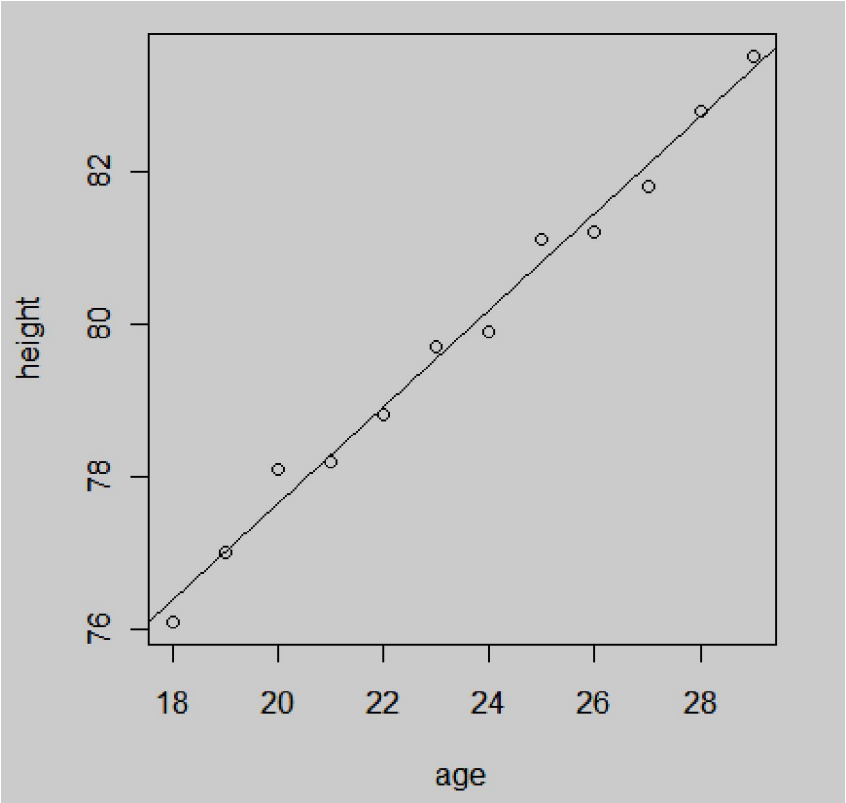
Kết quả trả về ta thu được giá trị chặn (intercept) $\alpha = 64.928$ và độ dốc $\beta = 0.635$, phương trình hồi quy là : $height = 0.635\ age + 64.928$.

Chúng ta có thể dùng hàm **abline()** để vẽ lên đồ thị phương trình hồi quy tuyến tính (hình 12.7).

```
abline(res)
```

Khi muốn dự đoán chiều cao của trẻ có tuổi là 23.5 tháng, chúng ta sử dụng phương trình hồi quy tìm được (*height = 0.635 age + 64.928*) để tính chiều cao là 79.85 cm. Việc này được thực hiện trong R bằng lệnh sau đây:

```
new <- data.frame(age = 23.5)
pred <- predict(res, new)
```



Hình 12.7: *Đồ thị của phương trình hồi quy về chiều cao trung bình của trẻ theo tháng tuổi*

12.4 LẬP TRÌNH TRONG MÔI TRƯỜNG R

Một tính năng rất quan trọng trong môi trường **R** là việc hỗ trợ lập trình. Người sử dụng có thể định nghĩa các hàm để giải quyết các yêu cầu đặt ra. Cú pháp cho định nghĩa một hàm trong **R** như sau:

```
tên_hàm <- function(danh_sách_tham_số)  {
        thân_hàm
}
```

Ví dụ theo sau minh họa cách viết hàm tên **desc()** nhận tham số đầu vào **x** là dãy số trả về kết quả bao gồm giá trị trung bình và độ lệch chuẩn của dãy số.

```
desc <- function(x) {
        mu <- mean(x)
        s <- sd(x)
        c(MEAN = mu, SD = s)
}
```

Để sử dụng hàm **desc()**, ta sinh dãy gồm 10 số nguyên ngẫu nhiên có giá trị từ 1 đến 100, sau đó truyền cho hàm **desc()**, kết quả lưu vào trong biến **d**, phần tử **d[1]** chứa giá trị trung bình, **d[2]** là độ lệch chuẩn.

```
x <- sample(1:100, 10, replace = FALSE)
d <- desc(x)
```

Một vài cấu trúc điều khiển cũng được sử dụng trong **R** như:

```
if (điều_kiện)  {
        công_việc_1
} else {
        công_việc_2
}

for (biến       in     khoảng_giá_trị) {
        công_việc
}

while (điều_kiện)  {
        công_việc
}

repeat {
        công_việc
}
```

Ví dụ tiếp theo định nghĩa một hàm tên **myfunc()** cho phép đọc một tập tin dữ liệu tên "**mat.dat**" lưu dạng ma trận, ngăn cách giữa các giá trị là ký tự khoảng trắng, sau đó chuẩn hóa dữ liệu trong khoảng [0, 1] và vẽ đồ thị cho từng cặp biến dữ liệu. Dòng lệnh 1 định nghĩa hàm **myfunc()** nhận tham số đầu vào là tập tin dữ liệu **infile**. Dòng lệnh 2 cho phép đọc tập tin **infile** (lưu dữ liệu dạng bảng, giá trị cách nhau bởi khoảng trắng) vào biến **tab**. Dòng lệnh 3 chuyển đổi kiểu bảng của **tab** thành ma trận. Sau đó dòng lệnh 4 cho phép tính số thuộc tính (cột) của **tab**. Dòng lệnh từ 5 đến 12 là vòng lặp thực hiện chuẩn hóa dữ liệu cho tất cả các thuộc tính. Dòng lệnh 13 cho phép hiển thị dữ liệu trên ma trận scatteplot 2 chiều. Dòng 15 thực hiện việc gọi hàm **myfunc()** để chuẩn hóa dữ liệu trong tập tin **mat.dat**.

Bảng 12.7: Ví dụ minh họa chuẩn hóa dữ liệu

```
Dòng  Mã lệnh
1     myfunc <- function(infile) {
2             tab <- read.table(infile)
3             tab <- as.matrix(tab)
4             ncol <- length(tab[1,])
5             for(i in 1:ncol) {
6                     min <- min(tab[,i])
7                     max <- max(tab[,i])
8                     rang <- max - min
9                     if (rang == 0)          # min == max
10                            rang <- 1
11                    tab[,i] <- (tab[,i] - min)/rang
12            }
13            pairs(tab, main = " My Plot ", pch = 21, col = "red")
14    }

15    myfunc("mat.data")
```

Với phần trình bày ngắn gọn cho thấy được rằng **R** rất dễ học và có thể phát triển nhanh các ứng dụng tính toán, xác suất thống kê, phân tích dữ liệu. Nếu độc giả quan tâm khả năng của **R**, nên tham khảo thêm các tài liệu [4, 5, 7, 9, 10].

BÀI TẬP

Anh, Chị đọc và viết lại toàn bộ ví dụ minh họa của phần ngôn ngữ **R**, hãy quan sát các kết quả thực thi.

1. Chiều cao trung bình hiện nay ở nam thanh niên là 160 cm, với độ lệch chuẩn là 4.6 cm. Giả sử rằng chiều cao này tuân theo luật phân phối chuẩn. Anh, Chị hãy xây dựng một hàm phân phối chiều cao cho toàn bộ quần thể nam thanh niên, vẽ lên đồ thị.

2. Qua theo dõi nhiều tháng về tình hình người đến gọi điện thoại tại các buồng điện thoại công cộng thì biết được, tính trung bình cứ khoảng 2000 người đi qua thì có 2 người ghé qua gọi điện thoại. Anh, Chị hãy cho biết xác suất để có hơn 2 người ghé qua gọi điện thoại công cộng là bao nhiêu.

3. Bằng **R**, Anh, Chị hãy thực hiện các công việc sau:

- Tạo dãy số nguyên từ 1 đến 100 lưu vào biến **d**.
- Lấy mẫu ngẫu nhiên 100 phần tử có hoàn lại từ tập **d** lưu vào biến **b**.
- Tính giá trị trung bình, phương sai của tập **b**.
- Lấy ra các phần tử có trong tập **d** nhưng không nằm trong tập **b**.
- Có bao nhiêu phần tử khác nhau nằm trong tập **b**.
- Vẽ đồ thị hộp và tổ chức đồ của tập **b**.

4. Giả sử nghiên cứu sự ảnh hưởng của các môn học toán, lý, hóa đến kết quả môn giải thuật được cho trong bảng như sau:

STT	Toán	Lý	Hóa	Giải thuật
1	8	6	5	7
2	5	5	5	5
3	7	6	5	6
4	6	5	5	5
5	9	5	6	9
6	10	6	5	9
7	5	7	6	4
8	6	8	5	5
9	7	6	6	6
10	7	5	5	7

11	7	6	6	6
12	6	7	5	4
13	6	8	6	5
14	5	7	5	4
15	5	5	6	4
16	10	6	5	10
17	9	8	5	8
18	9	7	5	9
19	9	6	5	9
20	9	8	5	10

Bằng **R**, Anh, Chị hãy hiển thị bảng dữ liệu trên với phương pháp Scatterplot 2 chiều và trục tọa độ song song. Tiếp đến là xây dựng mô hình hồi quy tuyến tính và dự báo điểm giải thuật của các sinh viên sau đây:

STT	**Toán**	**Lý**	**Hóa**	**Giải thuật**
21	5	5	6	?
22	7	6	5	?
23	6	6	8	?
24	10	5	5	?
25	9	5	9	?

TÀI LIỆU THAM KHẢO

1. Becker R.A., Chambers J.M. and Wilks A.R.: *The New S Language: A Programming Environment for Data Analysis and Graphics*. Chapman & Hall, 1988.
2. Burns P.: An Introduction to the S Language. 2002.
3. Carr D., Littlefield R., Nicholson W., and Littlefield J.: Scatterplot matrix techniques for large N. *Journal of the American Statistical Association* 82(398):424-436, 1987.
4. Crawley M.J.: *Statistics: An Introduction using R*. Wiley, 2005.

5. Ihaka R. and Gentleman R.: R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3): 299-314, 1996.
6. Inselberg A.: The Plane with Parallel Coordinates. in *Special Issue on the Computational Geometry of The Visual Computer*, vol. 1, n° 2, 1985, p. 69-97.
7. Maindonald J. and Braun J.: *Data Analysis and Graphics Using R*. Cambridge University Press, 2003.
8. Spector P.: *An Introduction to S and S-Plus*. Duxbury Press, 1994.
9. Spector P.: An Introduction to R. Statistical Computing Facility, University of California, Berkeley, 2004.
10. Venables W.N. and Smith D.M.: *An Introduction to R*. Network Theory, 2002.