

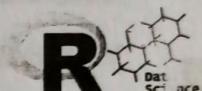
R for Data Science

Bài 17: *Linear Regression*

Phòng LT & Mạng

https://csc.edu.vn/lap-trinh-va-cSDL/R-Programming-Language-for-Data-Science_190

2020



Nội dung



1. Giới thiệu

2. Simple Linear Regression

3. Multiple Linear Regression

4. Model Selection

Giới thiệu

- Phân tích hồi quy (Regression analysis) là một công việc sử dụng công cụ thống kê để thiết lập mô hình quan hệ (relationship model) giữa hai biến.
- Một biến được gọi là predictor variable có giá trị được tập hợp thông qua các thí nghiệm. Biến còn lại được gọi là response variable có giá trị được tính từ predictor variable.



Giới thiệu

- Linenear regression là một thuật toán thuộc nhóm Supervised Learning được sử dụng cho regression.
- Là một trong những mô hình kỹ thuật được sử dụng rộng rãi
- Là một trong những chủ đề đầu tiên cần biết khi học về mô hình tiên đoán dữ liệu (learning predictive modeling).



Giới thiệu

❑ Ví dụ

- Hệ thống quản lý rủi ro hoặc chấm điểm tín dụng của các ngân hàng: để quyết định hạn mức thẻ Credit card của khách hàng, hệ thống sẽ sử dụng LR để tính toán dựa trên các thông tin khách hàng cung cấp như mức lương hiện tại, độ tuổi, giới tính, công việc đang làm v.v...
- Dự báo tài chính
- Dự đoán chi phí phần mềm, chi phí đảm bảo chất lượng phần mềm

Giới thiệu

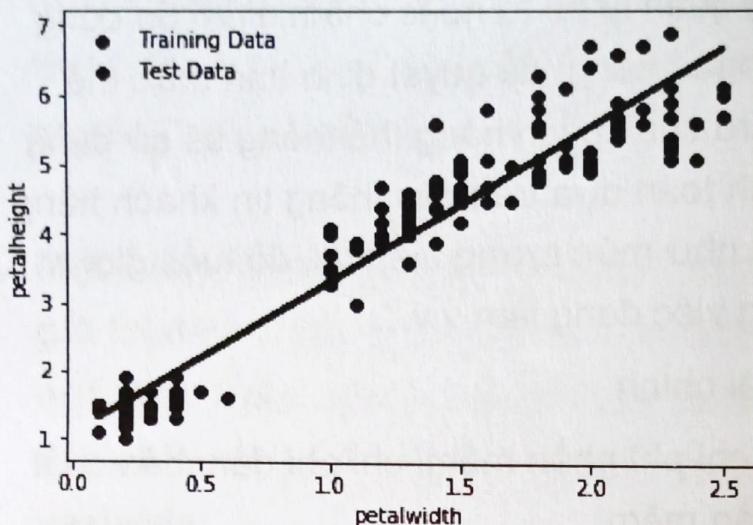
❑ Ý tưởng



- Tìm mối quan hệ giữa numerical output và input variables.
- Dependent variable liên tục, còn các independent variable có thể liên tục hoặc rời rạc
- Mối quan hệ được mô hình hóa dưới dạng tuyến tính (linear)
- Mô hình có thể được “phù hợp” bằng cách sử dụng least squares

Giới thiệu

□ Mô hình



Task: cho
petalwidth
=> tính
petalheight

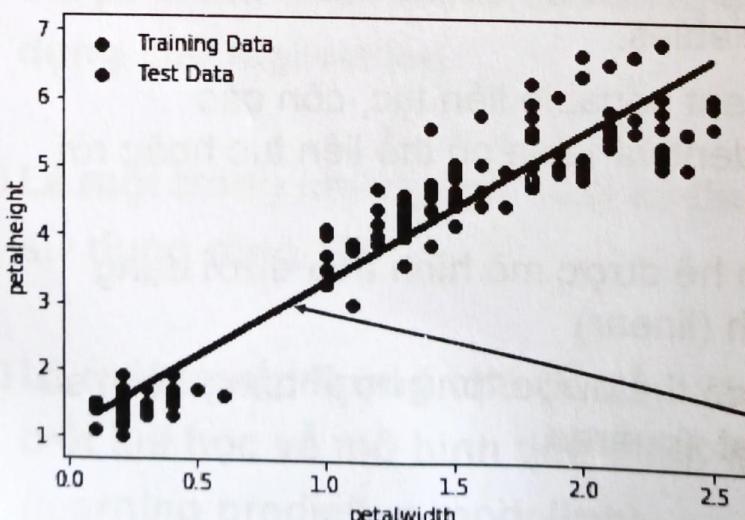


R programming language for Data Science

7

Giới thiệu

□ Mô hình



Regression
line

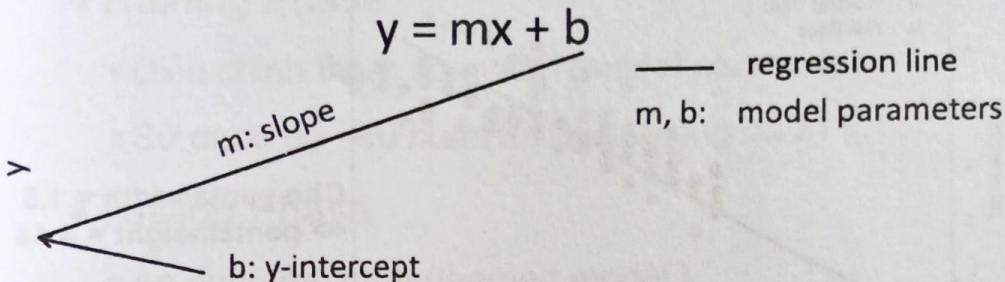


R programming language for Data Science

8

Giới thiệu

□ Least Squares Algorithm



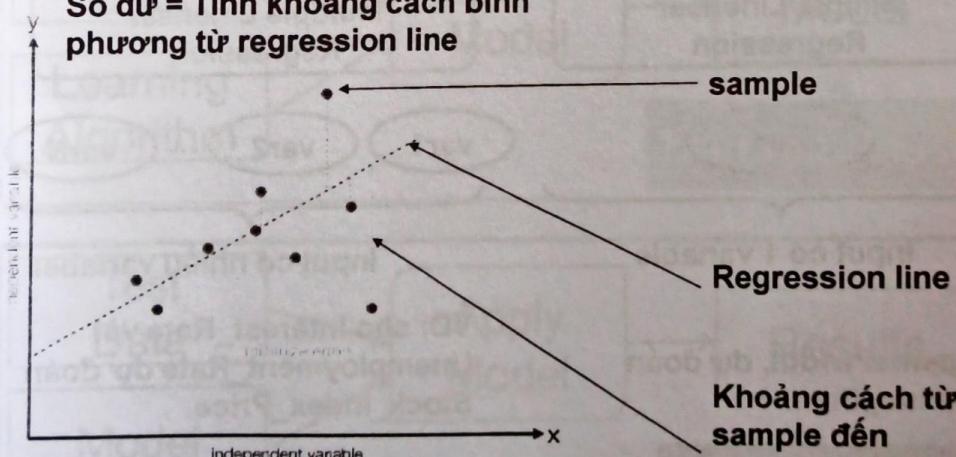
Mô hình huấn luyện linear regression điều chỉnh model parameter cho phù hợp với mẫu



Giới thiệu

□ Least Squares Method

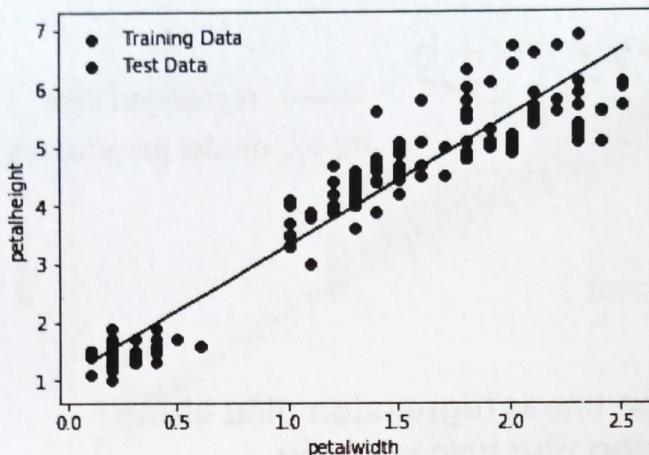
Số dư = Tính khoảng cách bình phương từ regression line



Mục tiêu: tìm regression line sao cho tổng số dư càng nhỏ càng tốt

Giới thiệu

□ Áp dụng mô hình



R programming language for Data Science

11

Giới thiệu

□ Phân loại

Simple Linenear
Regression

var1

Input có 1 variable

VD:

Cho petal width, dự đoán
petal length

Multiple Linenear
Regression

var1 var2 ... varN

Input có nhiều variabel

VD: cho Interest_Rate và

Unemployment_Rate dự đoán
Stock_Index_Price

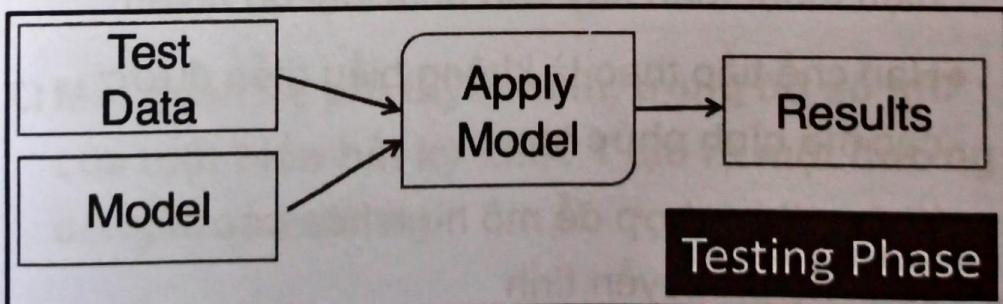
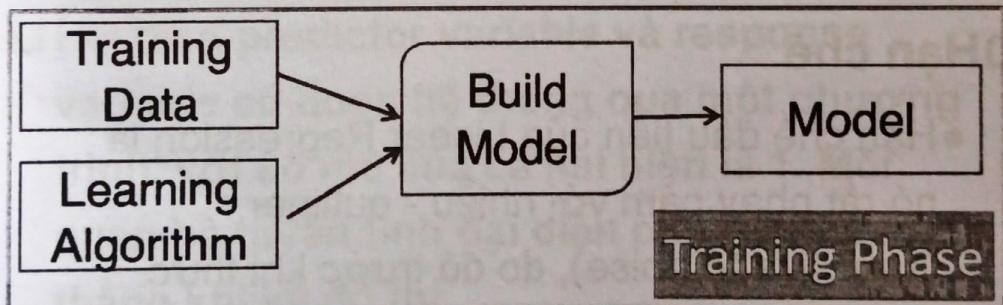
□ Xây dựng và áp dụng model

● Training Phase

- Điều chỉnh tham số model (model parameter)
- Sử dụng dữ liệu huấn luyện (training data)

● Testing Phase

- Áp dụng mô hình (learned model)
- Sử dụng dữ liệu mới (new data)



Giới thiệu

□ Ưu điểm

- Thuật toán dễ hiểu
- Độ phổ biến cao
- Tốc độ giải thuật nhanh
- Thực hiện mô hình thống kê, khi các mối quan hệ giữa các biến độc lập và biến phụ thuộc gần như tuyến tính => cho thấy kết quả tối ưu.



Giới thiệu

□ Hạn chế

- Hạn chế đầu tiên của Linear Regression là nó rất nhạy cảm với nhiễu - outlier (sensitive to noise), do đó trước khi thực hiện thuật toán này cần phải loại bỏ nhiễu
- Hạn chế tiếp theo là không biểu diễn được các mô hình phức tạp.
- Không thích hợp để mô hình hóa các mối quan hệ phi tuyến tính



Nội dung

1. Giới thiệu
2. Simple Linear Regression
3. Multiple Linear Regression
4. Model Selection



Simple Linear Regression

- Hai biến predictor variable và response variable có quan hệ thông qua một phương trình, với số mũ của cả hai biến là 1. Mỗi quan hệ tuyến tính đại diện cho một đường thẳng khi vẽ đồ thị.**
- Mỗi quan hệ phi tuyến tính, trong đó số mũ của một biến bất kỳ khác 1 tạo ra một đường cong khi vẽ đồ thị.**



Simple Linear Regression

□ Phương trình toán học: $y = m^*x + b$

- Trong đó:

- y : response variable
- x : predictor variable
- m (slope) và b (y _intercept) là những hằng số được gọi là hệ số (coefficient).



Simple Linear Regression

□ Các bước thực hiện

- Tìm hiểu về dataset (Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ liệu)
- Tạo training data và test data
- Xây dựng mô hình (model) sử dụng `lm(fomular, data = trainingData)` với training data
- Sử dụng mô hình để dự đoán cho test data dùng `predict(model, testData)`
- In `summary` của model, lấy giá trị các hệ số m , b
- Dự đoán kết quả cho các sample mới theo m , b hoặc theo `predict()`
- Trực quan hóa kết quả

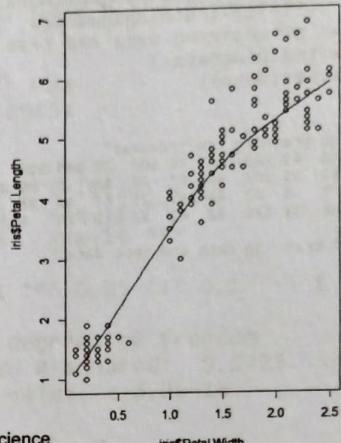


Linenear regression

- Tìm hiểu về dataset (Thực hiện thí nghiệm => Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ

```
# dataset understanding
head(iris)
# scatterplot
scatter.smooth(x=iris$Petal.Width, y=iris$Petal.Length, main="width ~ Length")
Width ~ Length
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

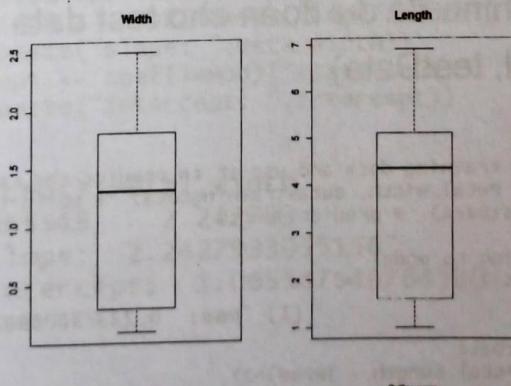


R programming language for Data Science

21

Linenear regression

```
# BoxPlot to check for outliers
par(mfrow=c(1, 2)) # divide graph area in 2 columns
boxplot(iris$Petal.Width, main="width",
        sub=paste("outlier rows: ", boxplot.stats(iris$Petal.Width)$out)) # box plot for 'width'
boxplot(iris$Petal.Length, main="Length",
        sub=paste("outlier rows: ", boxplot.stats(iris$Petal.Length)$out)) # box plot for 'Length'
```



```
# calculate correlation between width and Length
print(cor(iris$Petal.Width, iris$Petal.Length))
[1] 0.9628654
```

R programming language for Data Science

22

Linenear regression

- Tạo training data và test data

```
# Create the training (development) and test (validation) data samples from original data.
set.seed(42) # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(iris), 0.7*nrow(iris)) # row indices for training data
print("Selected training row indexes:")
print(trainingRowIndex)
trainingData <- iris[trainingRowIndex, ] # training data
testData <- iris[-trainingRowIndex, ] # test data
print("Rows of training data and test data:")
print(nrow(trainingData))
print(nrow(testData))

[1] "Selected training row indexes:"
[1] 138 140 43 123 94 76 107 20 146 100 65 141 129 35 63 127 132 16 136 74 118 18 135 121 11 149
[27] 49 112 55 102 89 97 46 81 1 96 116 24 128 68 42 48 5 105 130 101 93 66 139 134 34 137
[53] 40 77 4 72 64 133 25 47 61 88 67 50 131 17 23 69 57 143 115 12 85 37 104 54 114 28
[79] 38 86 41 126 92 44 52 119 15 6 144 19 70 71 13 75 103 60 150 73 39 32 51 83 124 90
[105] 120
[1] "Rows of training data and test data:"
[1] 105
[1] 45
```



Linenear regression

- Xây dựng mô hình (model) sử dụng lm(formular, data = trainingData) với training data
- Sử dụng mô hình để dự đoán cho test data dùng predict(model, testData)

```
# Develop the model on the training data and use it to predict the Length on test data
lmMod <- lm(Petal.Length ~ Petal.Width, data=trainingData) # build the model
iPred <- predict(lmMod, testData) # predict length

# mean square error according to model
mse <- mean(lmMod$residuals^2)
print(paste("mse: ", mse)) [1] "mse: 0.233736069922266"

# mean square error of testData
mse_test = mean((testData$Petal.Length - iPred)^2)
print(paste("mse in test: ", mse_test)) [1] "mse test: 0.2070617146726"
```



Linenear regression

- In summary của model, lấy giá trị các hệ số m, b

```
# Review diagnostic measures.
print(summary(lmMod)) # model summary

Call:
lm(formula = Petal.Length ~ Petal.Width, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.34825 -0.30258 -0.01411  0.24310  1.39454 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.06555   0.09187 11.60 <2e-16 ***
Petal.Width 2.24279   0.06374 35.19 <2e-16 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4881 on 103 degrees of freedom
Multiple R-squared:  0.9232, Adjusted R-squared:  0.9225 
F-statistic: 1238 on 1 and 103 DF,  p-value: < 2.2e-16
```

Linenear regression

- In summary của model, lấy giá trị các hệ số m, b

```
# model coefficients
print(coef(lmMod))
# get beta estimate for height
beta_width <- coef(lmMod)[["Petal.Width"]]
print(paste("slope: ",beta_width))
Intercept <- coef(lmMod)[["(Intercept)"]]
print(paste("Intercept: ",Intercept))

(Intercept) Petal.Width
1.065548 2.242793
[1] "slope: 2.2427933955156"
[1] "Intercept: 1.06554794876458"
```

Linenear regression

- Dự đoán kết quả cho các sample mới theo m, b hoặc theo predict

```
# new predictions
# solution 1
x <- c(0.9, 1.5, 2.1)
y <- Intercept + beta_speed * x
print("Solution 1 - results:")
print(y)
# solution 2
y1 <- predict(lmMod, data.frame(Petal.Width = x))
print("Solution 2 - results:")
print(y1)

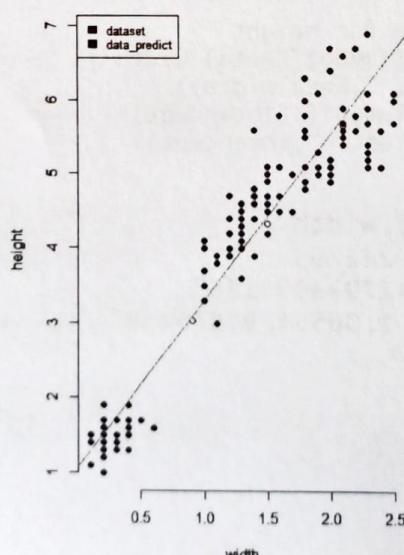
[1] "Solution 1 - results:"
[1] 3.030681 4.340770 5.650859
[1] "Solution 2 - results:"
      1       2       3
3.084062 4.429738 5.775414
```



Linenear regression

- Trực quan hóa kết quả

Iris Linear regression



Nội dung

1. Giới thiệu
2. Simple Linear Regression
3. Multiple Linear Regression
4. Model Selection



Multiple Linenear regression

- Là mở rộng của linear với mối quan hệ nhiều hơn 2 biến: nhiều predictor variable và một response variable
- Phương trình toán học: $y = m_1 * x_1 + m_2 * x_2 + \dots + m_n * x_n + b$
- Trong đó:
 - y: response variable
 - x_1, x_2, \dots, x_n : các predictor variable
 - m_1, m_2, \dots, m_n và b là những hằng số được gọi là hệ số (coefficient).



Multiple Linenear regression

□ Các bước thực hiện

- Tìm hiểu về dataset (Thực hiện thí nghiệm => Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ liệu)
- Tạo training data và test data
- Xây dựng mô hình (model) sử dụng lm(fomular, data = trainingData) với training data
- Sử dụng mô hình để dự đoán cho test data dùng predict(model, testData)
- In summary của model, lấy giá trị các hệ số m, b₁, b₂, ..., b_n
- Dự đoán kết quả cho các sample mới theo m, b hoặc theo predict()
- Trực quan hóa kết quả

R programming language for Data Science

31

Multiple Linenear regression

- Tìm hiểu về dataset (Thực hiện thí nghiệm => Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ liệu)

```
# dataset understanding
head(mtcars)
# get dataset columns for prediction
input <- mtcars[,c("mpg","disp","hp","wt")]
head(input)
# visualization
pairs(~mpg+disp+hp+wt, data = input)
```

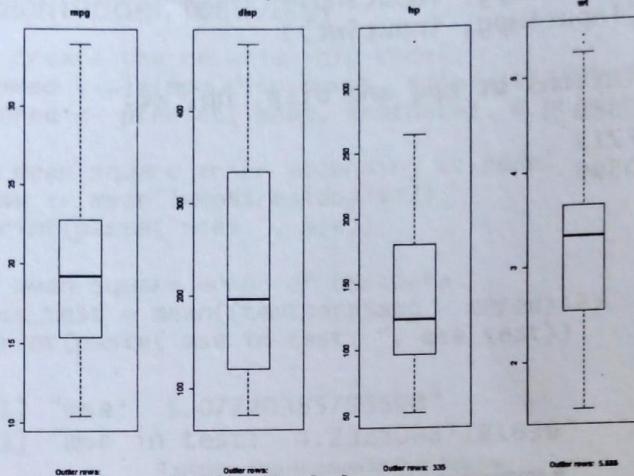
	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	4
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1
	mpg	disp	hp	wt							
Mazda RX4	21.0	160	110	2.620							
Mazda RX4 Wag	21.0	160	110	2.875							
Datsun 710	22.8	108	93	2.320							
Hornet 4 Drive	21.4	258	110	3.215							
Hornet Sportabout	18.7	360	175	3.440							
Valiant	18.1	225	105	3.460							

R programming language for Data Science

32

Multiple Linenear regression

```
# BoxPlot to check for outliers
par(mfrow=c(1, 4)) # divide graph area in 4 columns
boxplot(input$mpg, main="mpg",
        sub=paste("Outlier rows: ", boxplot.stats(input$mpg)$out)) # box plot for 'mpg'
boxplot(input$disp, main="disp",
        sub=paste("Outlier rows: ", boxplot.stats(input$disp)$out)) # box plot for 'disp'
boxplot(input$hp, main="hp",
        sub=paste("Outlier rows: ", boxplot.stats(input$hp)$out)) # box plot for 'hp'
boxplot(input$wt, main="wt",
        sub=paste("Outlier rows: ", boxplot.stats(input$wt)$out)) # box plot for 'wt'
```



33

Multiple Linenear regression

- Xử lý các dòng chứa outlier (loại bỏ dòng chứa outlier, cập nhật giá trị)

```
hp_outliers <- boxplot.stats(input$hp)$out
print("hp_outliers: ")
print(hp_outliers)
print(paste("Numrows: ", sum(input$hp == hp_outliers[1])))

wt_outliers <- c(boxplot.stats(input$wt)$out)
print("wt_outliers: ")
print(wt_outliers)
print(paste("Numrows: ", sum(input$wt == wt_outliers[1])+sum(input$wt == wt_outliers[2])))

#drop rows have outliers
print(paste("Before drop:", nrow(input)))
input <- input[input$hp != hp_outliers[1],]
input <- input[input$wt != wt_outliers[1],]
input <- input[input$wt != wt_outliers[2],]
print(paste("After drop:", nrow(input)))

[1] "hp_outliers: "
[1] 335
[1] "Numrows: 1"
[1] "wt_outliers: "
[1] 5.424 5.345
[1] "Numrows: 2"
[1] "Before drop: 32"
[1] "After drop: 29"
```

Multiple Linenear regression



- Xem xét mối quan hệ giữa các biến

```
# calculate correlation between mpg and disp, hp, wt
print("Correlations bt mpg and disp, hp, wt:")
print(cor(input$mpg, input$disp))
print(cor(input$mpg, input$hp))
print(cor(input$mpg, input$wt))

[1] "Correlations bt mpg and disp, hp, wt:"
[1] -0.8290389
[1] -0.7997213
[1] -0.8830596
```



Multiple Linenear regression



- Tạo training data và test data

```
# Create the training (development) and test (validation) data.
set.seed(42) # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(input), 0.8*nrow(input))
print("Selected training row indexes:")
print(trainingRowIndex)
trainingData <- input[trainingRowIndex, ] # training data
testData <- input[-trainingRowIndex, ] # test data
print("Rows of training data and test data:")
print(nrow(trainingData))
print(nrow(testData))

[1] "Selected training row indexes:"
[1] 27 29 8 22 17 13 25 3 14 15 9 24 16 5 7 21 18 2 6 11 19 12 20
[1] "Rows of training data and test data:"
[1] 23
[1] 6
```



Multiple Linenear regression



- Xây dựng mô hình (model) sử dụng lm(fomular, data = trainingData) với training data
- Sử dụng mô hình để dự đoán cho test data dùng predict(model, testData)

```
# Create the relationship model.  
lmMod <- lm(mpg~disp+hp+wt, data = trainingData)  
cPred <- predict(lmMod, testData) # predict mpg  
  
# mean square error according to model  
mse <- mean(lmMod$residuals^2)  
print(paste("mse: ", mse))  
  
# mean square error of testData  
mse_test = mean((testData$mpg - cPred)^2)  
print(paste("mse in test: ", mse_test))  
  
[1] "mse: 5.07720385795598"  
[1] "mse in test: 4.23260887181639"
```

R programming language for Data Science

37

Multiple Linenear regression



- In summary của model, lấy giá trị các hệ số $m_1, m_2 \dots, b$

```
# Show the model.  
print(summary(model))  
  
Call:  
lm(formula = mpg ~ disp + hp + wt, data = input)  
  
Residuals:  
    Min      1Q  Median      3Q     Max  
-3.891 -1.640 -0.172  1.061  5.861  
  
Coefficients:  
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) 37.105505   2.110815 17.579 < 2e-16 ***  
disp        -0.000937   0.010350 -0.091  0.92851  
hp          -0.031157   0.011436 -2.724  0.01097 *  
wt         -3.800891   1.066191 -3.565  0.00133 **  
---  
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 2.639 on 28 degrees of freedom
Multiple R-Squared: 0.8268, Adjusted R-squared: 0.8083
F-statistic: 44.57 on 3 and 28 DF, p-value: 8.65e-11

R programming language for Data Science

38

Multiple Linenear regression

- In summary của model, lấy giá trị các hệ số m, b

```
# Get the Intercept and coefficients as vector elements.
cat("# # # # The Coefficient values # # # ", "\n")

b <- coef(model)[1]
print(b)

mdisp <- coef(model)[2]
mhp <- coef(model)[3]
mwt <- coef(model)[4]

print(mdisp)      # # # # The coefficient values # # #
print(mhp)        (Intercept)
print(mwt)        37.10551
                  disp
                  -0.0009370091
                  hp
                  -0.03115655
                  wt
                  -3.800891
```

R programming language for Data Science

39



Multiple Linenear regression

- Dự đoán kết quả cho các sample mới theo m, b hoặc
theo predict

```
# new predictions
#disp = 221, hp = 102 and wt = 2.91
x1 <- 221
x2 <- 102
x3 <- 2.91

y = (mdisp*x1 + mhp*x2 + mwt*x3 + b)
print("Solution 1 - results:")
print(y)

# solution 2
y1 <- predict(lmMod, data.frame(disp = x1, hp = x2, wt = x3 ))
print("Solution 2 - results:")
print(y1)
```

[1] "Solution 1 - results:"

22.65987

[1] "Solution 2 - results:"

1

22.56627

R programming language for Data Science



40

1. Giới thiệu
2. Simple Linear Regression
3. Multiple Linear Regression
4. Model Selection

Model Selection

- ❑ Một trong những phương pháp lựa chọn mô hình hấp dẫn nhất là BMA (Bayesian Model Average)

- ❑ `install.packages('BMA')`

Model Selection

❑ Ví dụ: Cho dataset mtcars, hãy chọn một model phù hợp nhất để dự đoán mpg từ những thuộc tính còn lại.

`head(mtcars)`

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1



Model Selection

```
cols = colnames(mtcars)
cols

'mpg' 'cyl' 'disp' 'hp' 'drat' 'wt' 'qsec' 'vs' 'am' 'gear' 'carb'

yvar = mtcars[, ("mpg")]
xvars = mtcars[, cols[-1]]
bma = bicreg(xvars, yvar, strict = F, OR=2)
```



Model Selection



```
print(summary(bma))
```

```
call:  
bicreg(x = xvars, y = yvar, strict = F, OR = 2)
```

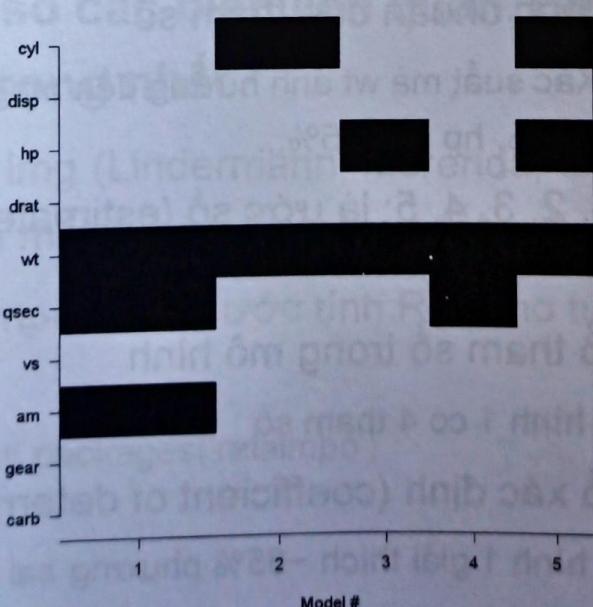
```
5 models were selected  
Best 5 models (cumulative posterior probability = 1):
```

	p!=0	EV	SD	model 1	model 2	model 3	model 4	model 5
Intercept	100.0	27.211564	13.82229	9.61778	39.68626	37.22727	19.74622	38.75179
cyl	37.9	-0.489171	0.71049	.	-1.50779	.	.	-0.94162
disp	0.0	0.000000	0.00000
hp	31.5	-0.008011	0.01373	.	.	-0.03177	.	-0.01804
drat	0.0	0.000000	0.00000
wt	100.0	-3.816159	0.93260	-3.91650	-3.19097	-3.87783	-5.04798	-3.16697
qsec	45.2	0.505065	0.59539	1.22589	.	.	0.92920	.
vs	0.0	0.000000	0.00000
am	28.8	0.845872	1.53016	2.93584
gear	0.0	0.000000	0.00000
carb	0.0	0.000000	0.00000
nvar				3	2	2	2	3
r2				0.850	0.830	0.827	0.826	0.843
BIC				-50.23818	-49.81447	-49.17255	-49.10426	-48.88168
post prob				0.288	0.233	0.169	0.163	0.146

Model Selection



Models selected by BMA



Model Selection

□ Diễn giải kết quả

- Cung cấp 10 thuộc tính, BMA tìm được 5 mô hình “phù hợp”
- $p!=0$: xác suất biến số có ảnh hưởng.
 - Ví dụ: Xác suất mà wt ảnh hưởng đến mpg là 100%, cyl là 37.9%, hp là 31.5%...
- EV (expected value): giá trị trung bình của thuộc tính.
 - Vd: Tương tự, mỗi đơn vị giảm wt có liên quan đến 3.8% tăng mpg

R programming language for Data Science

47

Model Selection

- **SD**: độ lệch chuẩn của tham số
 - Ví dụ: Xác suất mà wt ảnh hưởng đến mpg là 100%, cyl là 37.9%, hp là 31.5%...
- Model 1, 2, 3, 4, 5: là ước số (estimate) của tham số
- **nVar**: số tham số trong mô hình
 - Vd: Mô hình 1 có 4 tham số
- **r2**: hệ số xác định (coefficient of determination).
 - Vd: Mô hình 1 giải thích ~85% phương sai của mpg

R programming language for Data Science

48

- BIC: Bayesian Information Criterion, giá trị càng thấp, mô hình càng 'ưu tiên'
 - post prob (posterior probability): xác suất hậu định của mô hình.
 - Vd: Mô hình 1 xuất hiện 28.8% trong hàng nghìn lần tái chọn mẫu
- Không có mô hình "tốt nhất", chỉ có mô hình "phù hợp"
- "Phù hợp" = ít tham số + giải thích dữ liệu nhiều



❑ Trong số các biến liên quan, biến nào quan trọng nhất?

- Dùng Img (Lindermann, Merenda, Gold): đây là một thước đo mới và tốt
- Relaimpo có thể ước tính R^2 cho từng biến.
 - `install.packages('relaimpo')`



Model Selection

• Ví dụ

```
library("relaimpo")
```

```
m = lm(mpg ~ wt + qsec + am, data = mtcars)
```

```
calc.relimp(m, type="lmg", rela=T, rank=T)
```

Response variable: mpg
 Total response variance: 36.3241
 Analysis based on 32 observations

3 Regressors:

wt qsec am

Proportion of variance explained by model: 84.97%
 Metrics are normalized to sum to 100% (rela=TRUE).

Relative importance metrics:

	lmg
wt	0.5640407
qsec	0.1853428
R program am	0.2506165

51



Chapter 17: Linear Regression

Exercise 1: Baseball

Yêu cầu: Áp dụng Line Regression để dự đoán cân nặng dựa trên chiều cao

Cho dữ liệu baseball_2D.txt. Hãy áp dụng Line Regression để dự đoán cân nặng dựa trên chiều cao

- Đọc dữ liệu và gán cho biến data.
- Xem thông tin data: head(), số dòng, số cột, str, summary
- Chỉnh dữ liệu theo công thức: chiều cao (m) = chiều cao (inch) * 0.0254, cân nặng (kg) = cân nặng (pound) * 0.453592.
- Vẽ biểu đồ quan sát mối liên hệ giữa inputs và outputs data (scatter plot)
- Kiểm tra outliers => loại outliers
- Tạo train:test từ dữ liệu data với tỉ lệ 70:30
- Thực hiện Linear Regression với X_train, y_train.
- In summary của model
- Dự đoán y_pred từ test => so sánh với y_test
- Tính Mean Square Error (mse)
- Tính Coefficients, Intercept và Variance score
- Cho chiều cao lần lượt: x <- c(1.775, 1.825, 1.925) => dự đoán cân nặng
- Vẽ hình và xem kết quả

In [1]: # predict weight based on height

```
# open and read csv file
data <- read.csv("baseball.csv")
print(head(data))
print(is.data.frame(data))
print(paste("cols", ncol(data)))
print(paste("rows:", nrow(data)))
```

	Name	Team	Position	Height	Weight	Age	PosCategory
1	Adam_Donachie	BAL	Catcher	74	180	22.99	Catcher
2	Paul_Bako	BAL	Catcher	74	215	34.69	Catcher
3	Ramon_Hernandez	BAL	Catcher	72	210	30.78	Catcher
4	Kevin_Millar	BAL	First_Baseman	72	210	35.43	Infielder
5	Chris_Gomez	BAL	First_Baseman	73	188	35.71	Infielder
6	Brian_Roberts	BAL	Second_Baseman	69	176	29.39	Infielder

[1] TRUE

[1] "cols 7"

[1] "rows: 1015"



In [2]: summary(data)

	Name	Team	Position	Height
	Weight	Age	PosCategory	
Chris_Young	: 2	NYM : 38	Relief_Pitcher :315	Min. :67.00
Tony_Peña	: 2	ATL : 37	Starting_Pitcher:220	1st Qu.:72.00
A.J._Burnett	: 1	CHC : 36	Outfielder :194	Median :74.00
A.J._Murray	: 1	DET : 36	Catcher : 76	Mean :73.69
A.J._Pierzynski	: 1	OAK : 36	Second_Baseman : 58	3rd Qu.:75.00
Aaron_Boone	: 1	WAS : 36	First_Baseman : 55	Max. :83.00
(Other)	:1007	(Other):796	(Other) : 97	
Min.	:150.0	Min. :20.90	Catcher : 76	
1st Qu.	:186.0	1st Qu.:25.41	Infielder :210	
Median	:200.0	Median :27.90	Outfielder:194	
Mean	:201.3	Mean :28.71	Pitcher :535	
3rd Qu.	:215.0	3rd Qu.:31.19		
Max.	:290.0	Max. :48.52		

In [3]: str(data)

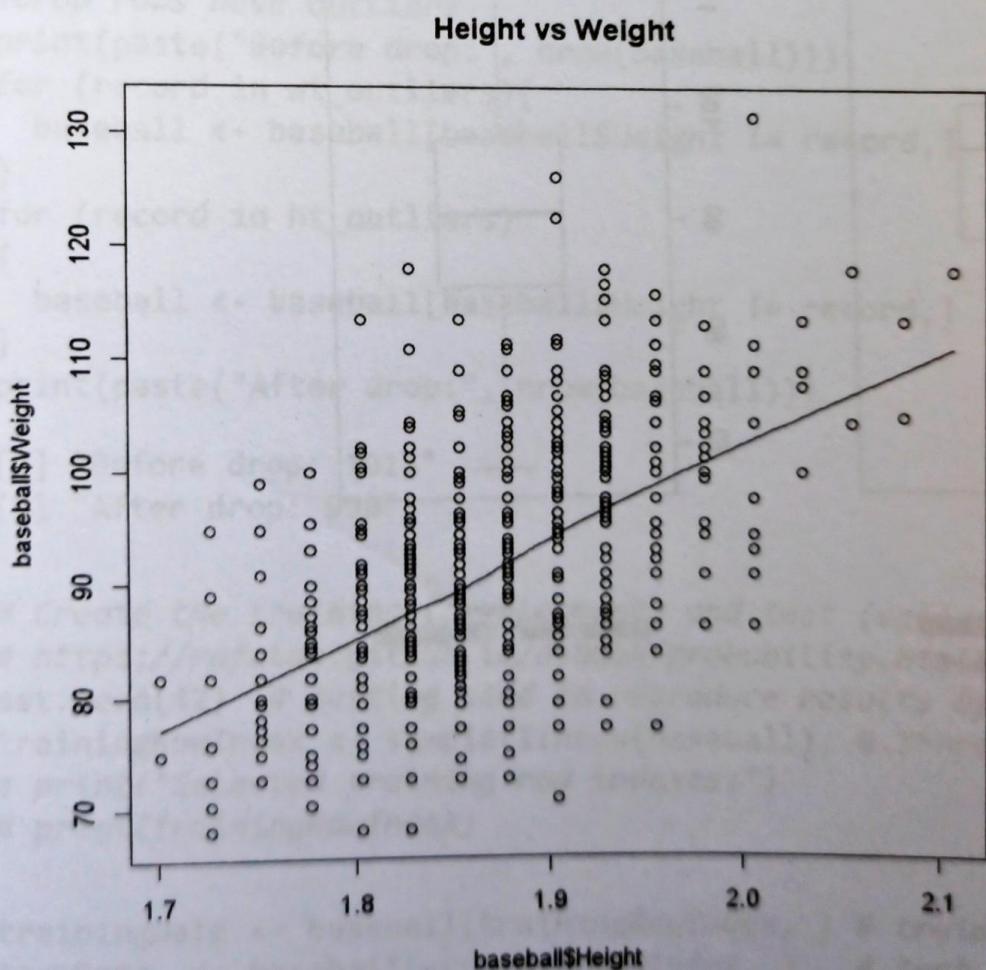
```
'data.frame': 1015 obs. of 7 variables:
 $ Name      : Factor w/ 1013 levels "A.J._Burnett",...: 13 778 801 615 199 134
 717 703 66 22 ...
 $ Team       : Factor w/ 30 levels "ANA","ARZ","ATL",...: 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 ...
 $ Position   : Factor w/ 8 levels "Catcher","First_Baseman",...: 1 1 1 2 2 5 6
 8 8 3 ...
 $ Height     : int  74 74 72 72 73 69 69 71 76 71 ...
 $ Weight     : int  180 215 210 210 188 176 209 200 231 180 ...
 $ Age        : num  23 34.7 30.8 35.4 35.7 ...
 $ PosCategory: Factor w/ 4 levels "Catcher","Infielder",...: 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 ...
```

In [4]: baseball <- data[c("Height", "Weight")]
print(head(baseball))

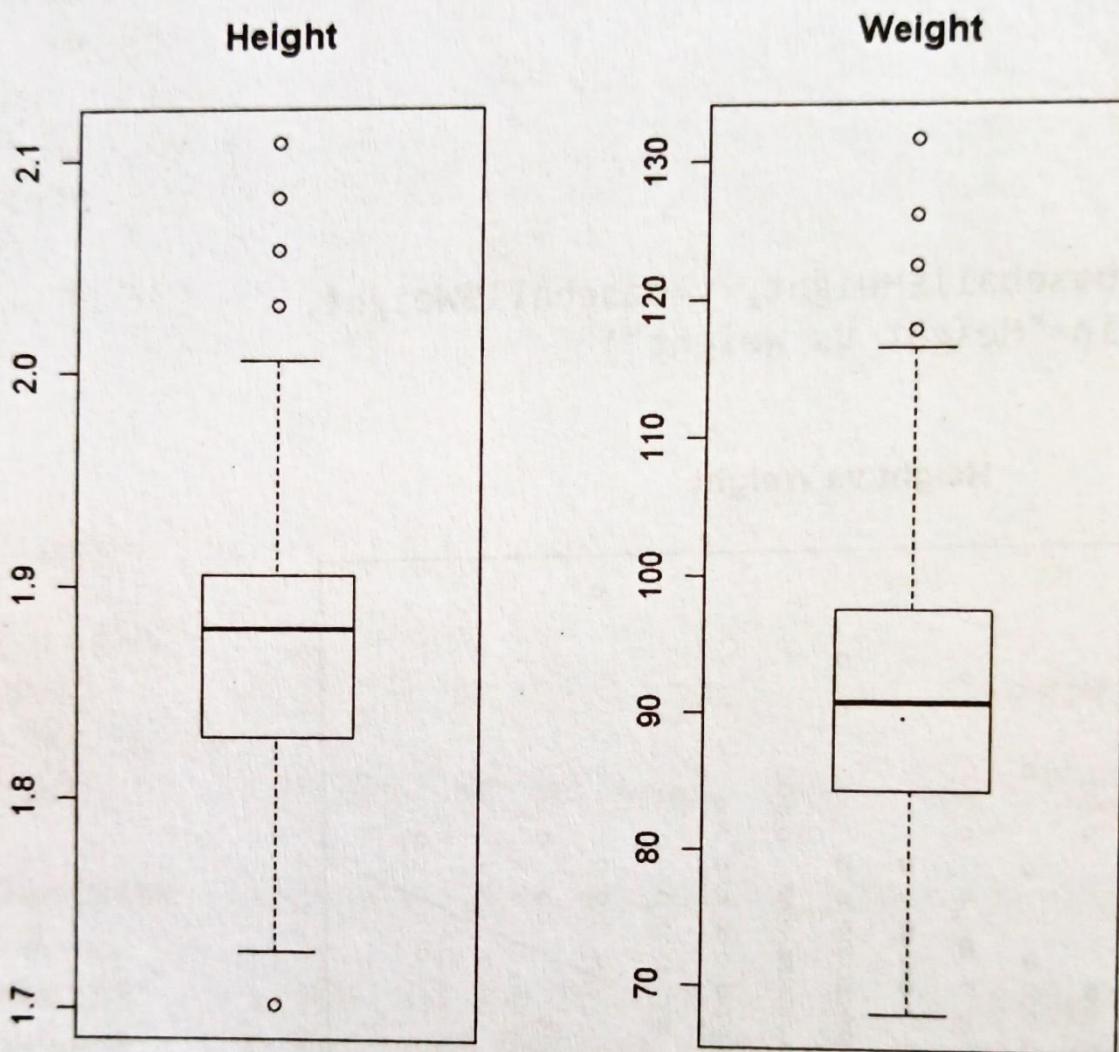
	Height	Weight
1	74	180
2	74	215
3	72	210
4	72	210
5	73	188
6	69	176

```
In [5]: baseball["Height"] <- baseball["Height"] * 0.0254  
baseball["Weight"] <- baseball["Weight"] * 0.453592  
  
print("After preprocessing data:")  
print(head(baseball))  
  
[1] "After preprocessing data:"  
  Height   Weight  
1 1.8796 81.64656  
2 1.8796 97.52228  
3 1.8288 95.25432  
4 1.8288 95.25432  
5 1.8542 85.27530  
6 1.7526 79.83219
```

```
In [6]: scatter.smooth(x=baseball$Height, y=baseball$Weight,  
                      main="Height vs Weight")
```



```
In [7]: # BoxPlot to Check for outliers
par(mfrow=c(1, 2)) # divide graph area in 2 columns
boxplot(baseball$Height, main="Height",
        sub=paste("Outlier rows: ",
                  boxplot.stats(baseball$Height)$out))
boxplot(baseball$Weight, main="Weight",
        sub=paste("Outlier rows: ",
                  boxplot.stats(baseball$Weight)$out))
```



Outlier rows: 200 201 202

Outlier rows: 123 124 125



```
In [3]: # calculate correlation between Width and Length
print(cor(baseball$Height, baseball$Weight))

wt_outliers <- c(boxplot.stats(baseball$Weight)$out)
print("wt_outliers: ")
print(wt_outliers)

ht_outliers <- c(boxplot.stats(baseball$Height)$out)
print("ht_outliers: ")
print(ht_outliers)

[1] 0.5315393
[1] "wt_outliers: "
[1] 117.9339 122.4698 131.5417 126.0986 117.9339 117.9339 117.9339
[1] "ht_outliers: "
[1] 2.0574 2.0320 2.0320 2.0320 2.0320 2.0828 2.0320 2.0574 2.0828 2.1082
[11] 1.7018 1.7018

In [9]: #drop rows have outliers
print(paste("Before drop:", nrow(baseball)))
for (record in wt_outliers){
  baseball <- baseball[baseball$Weight != record,]
}
for (record in ht_outliers)
{
  baseball <- baseball[baseball$Height != record,]
}
print(paste("After drop:", nrow(baseball)))

[1] "Before drop: 1015"
[1] "After drop: 998"

In [10]: # Create the training (development) and test (validation) data.
# https://rafalab.github.io/dsbook/probability.html#monte-carlo-simulations-for-
set.seed(42) # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(baseball), 0.7*nrow(baseball))
# print("Selected training row indexes:")
# print(trainingRowIndex)

In [11]: trainingData <- baseball[trainingRowIndex, ] # training data
 testData <- baseball[-trainingRowIndex, ] # test data

In [12]: print("Rows of training data and test data:")
print(nrow(trainingData))
print(nrow(testData))

[1] "Rows of training data and test data:"
[1] 698
[1] 300

In [13]: # Develop the model on the training data and use it to predict the Length on test
lmMod <- lm(Weight ~ Height, data=trainingData) # build the model
```

```
In [14]: iPred <- predict(lmMod, testData) # predict Length
          # mean square error according to model
          mse <- mean(lmMod$residuals^2)
          print(paste("mse: ", mse))

[1] "mse: 61.1770098543297"
```

```
In [15]: # mean square error of testData
          mse_test = mean((testData$Weight - iPred)^2)
          print(paste("mse in test: ", mse_test))

[1] "mse in test: 59.3539332937392"
```

```
In [16]: summary(lmMod)$r.squared
          # => r^2 has low value, this model fits ~ 27% data => not good!
          0.2687499653293
```

```
In [17]: # Review diagnostic measures.
          print(summary(lmMod)) # model summary
```

Call:

```
lm(formula = Weight ~ Height, data = trainingData)
```

Residuals:

Min	1Q	Median	3Q	Max
-21.3701	-5.8208	0.4391	5.3014	27.8722

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-67.775	9.937	-6.82	1.98e-11 ***
Height	85.006	5.315	15.99	< 2e-16 ***

Signif. codes:	0 ****	0.001 ***	0.01 **	0.05 *
	' . '	' . '	' . '	' . '

Residual standard error: 7.833 on 696 degrees of freedom

Multiple R-squared: 0.2687, Adjusted R-squared: 0.2677

F-statistic: 255.8 on 1 and 696 DF, p-value: < 2.2e-16

```
In [18]: # model coefficients
          print(coef(lmMod))
          # get beta estimate for height
          beta_height <- coef(lmMod)[["Height"]]
          print(paste("slope: ", beta_height))
          Intercept <- coef(lmMod)[["(Intercept)"]]
          print(paste("Intercept: ", Intercept))

          (Intercept)      Height
          -67.77469     85.00638
[1] "slope: 85.0063807905534"
[1] "Intercept: -67.7746921487327"
```

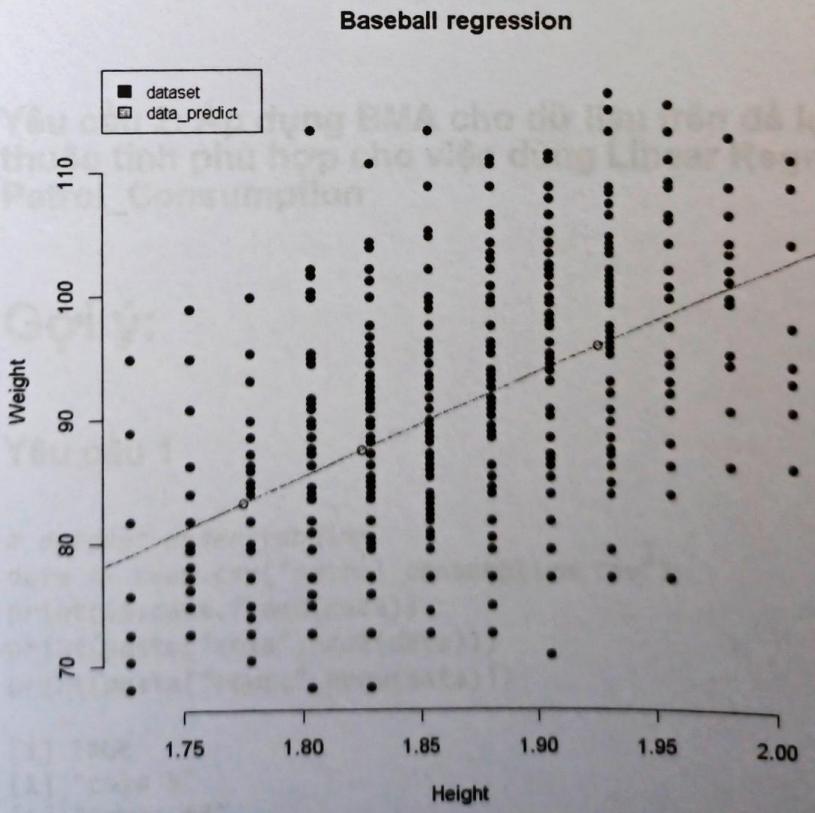
In [19]: # new predictions
solution 1
 $x \leftarrow c(1.775, 1.825, 1.925)$
 $y \leftarrow \text{Intercept} + \text{beta_height} * x$
print("Solution 1 - results:")
print(y)

[1] "Solution 1 - results:"
[1] 83.11163 87.36195 95.86259

In [20]: # solution 2
 $y1 \leftarrow \text{predict(lmMod, data.frame(Height = x))}$
print("Solution 2 - results:")
print(y1)

[1] "Solution 2 - results:"
1 2 3
83.11163 87.36195 95.86259

In [21]: # visualization
plot(baseball\$Height, baseball\$Weight,
main = "Baseball regression",
xlab = "Height", ylab = "Weight",
pch = 19, frame = FALSE, col = 'blue')
lines(x, y, col = 'red', type = 'p')
abline(lmMod, baseball, col = "green")
legend("topleft", c("dataset", "data_predict"),
cex = 0.8, fill = c("blue", "orange"))



Chapter 17: Linear Regression

Exercise 2: Petrol consumption

Yêu cầu 1: Áp dụng Line Regression để dự đoán Petrol_Consumption dựa trên Petrol_tax, Population_Driver_licence(%)

Cho dữ liệu petrol_consumption.csv. Hãy áp dụng Line Regression để dự đoán Petrol_Consumption dựa trên Petrol_tax, Population_Driver_licence(%)

- Đọc dữ liệu và gán cho biến data.
- Xem thông tin data: head(), số dòng, số cột, str, summary
- Vẽ biểu đồ quan sát mối liên hệ giữa Petrol_tax với Petrol_Consumption, Average_income với Petrol_Consumption, Paved_Highways với Petrol_Consumption, Population_Driver_licence(%) với Petrol_Consumption
- Kiểm tra outliers => loại outliers
- Tạo train:test từ dữ liệu data với tỉ lệ 80:20
- Thực hiện Linear Regression với train.
- In summary của model
- Dự đoán y_pred từ test => so sánh với y_test
- Tính Mean Square Error (mse)
- Tính Coefficients, Intercept và Variance score
- Nhận xét dựa trên kết quả

Yêu cầu 2: Áp dụng BMA cho dữ liệu trên để lựa chọn model với các thuộc tính phù hợp cho việc dùng Linear Regression dự đoán Petrol_Consumption

Gợi ý:

Yêu cầu 1

```
In [1]: # dataset understanding
data <- read.csv("petrol_consumption.csv")
print(is.data.frame(data))
print(paste("cols", ncol(data)))
print(paste("rows:", nrow(data)))

[1] TRUE
[1] "cols 5"
[1] "rows: 48"
```



In [2]: `print(head(data))`

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence...	
1	9.0	3571	1976		0.525
2	9.0	4092	1250		0.572
3	9.0	3865	1586		0.580
4	7.5	4870	2351		0.529
5	8.0	4399	431		0.544
6	10.0	5342	1333		0.571
	Petrol_Consumption				
1		541			
2		524			
3		561			
4		414			
5		410			
6		457			

In [3]: `str(data)`

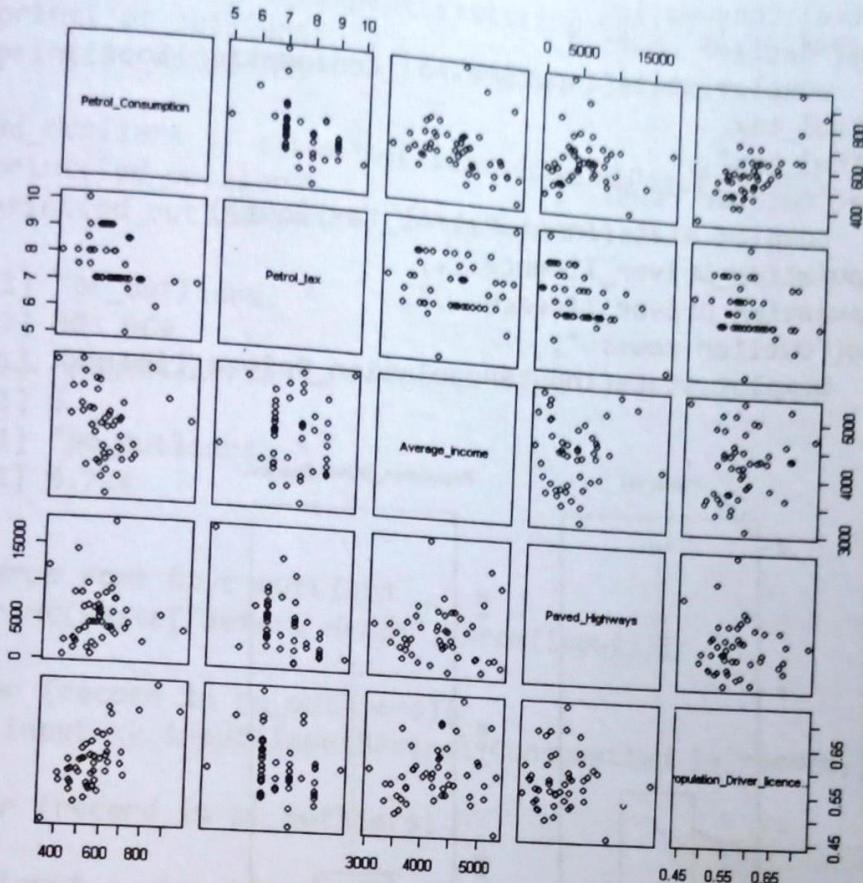
```
'data.frame': 48 obs. of 5 variables:
 $ Petrol_tax           : num  9 9 9 7.5 8 10 8 8 8 7 ...
 $ Average_income        : int  3571 4092 3865 4870 4399 5342 5319 5126 4
447 4512 ...
 $ Paved_Highways       : int  1976 1250 1586 2351 431 1333 11868 2138 8
577 8507 ...
 $ Population_Driver_licence...: num  0.525 0.572 0.58 0.529 0.544 0.571 0.451
0.553 0.529 0.552 ...
 $ Petrol_Consumption   : int  541 524 561 414 410 457 344 467 464 498
...

```

In [4]: `summary(data)`

Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence...
Min. : 5.000	Min. :3063	Min. : 431	Min. : 0.4510
1st Qu.: 7.000	1st Qu.:3739	1st Qu.: 3110	1st Qu.: 0.5298
Median : 7.500	Median :4298	Median : 4736	Median : 0.5645
Mean : 7.668	Mean :4242	Mean : 5565	Mean : 0.5703
3rd Qu.: 8.125	3rd Qu.:4579	3rd Qu.: 7156	3rd Qu.: 0.5952
Max. :10.000	Max. :5342	Max. :17782	Max. : 0.7240
Petrol_Consumption			
Min. :344.0			
1st Qu.:509.5			
Median :568.5			
Mean :576.8			
3rd Qu.:632.8			
Max. :968.0			

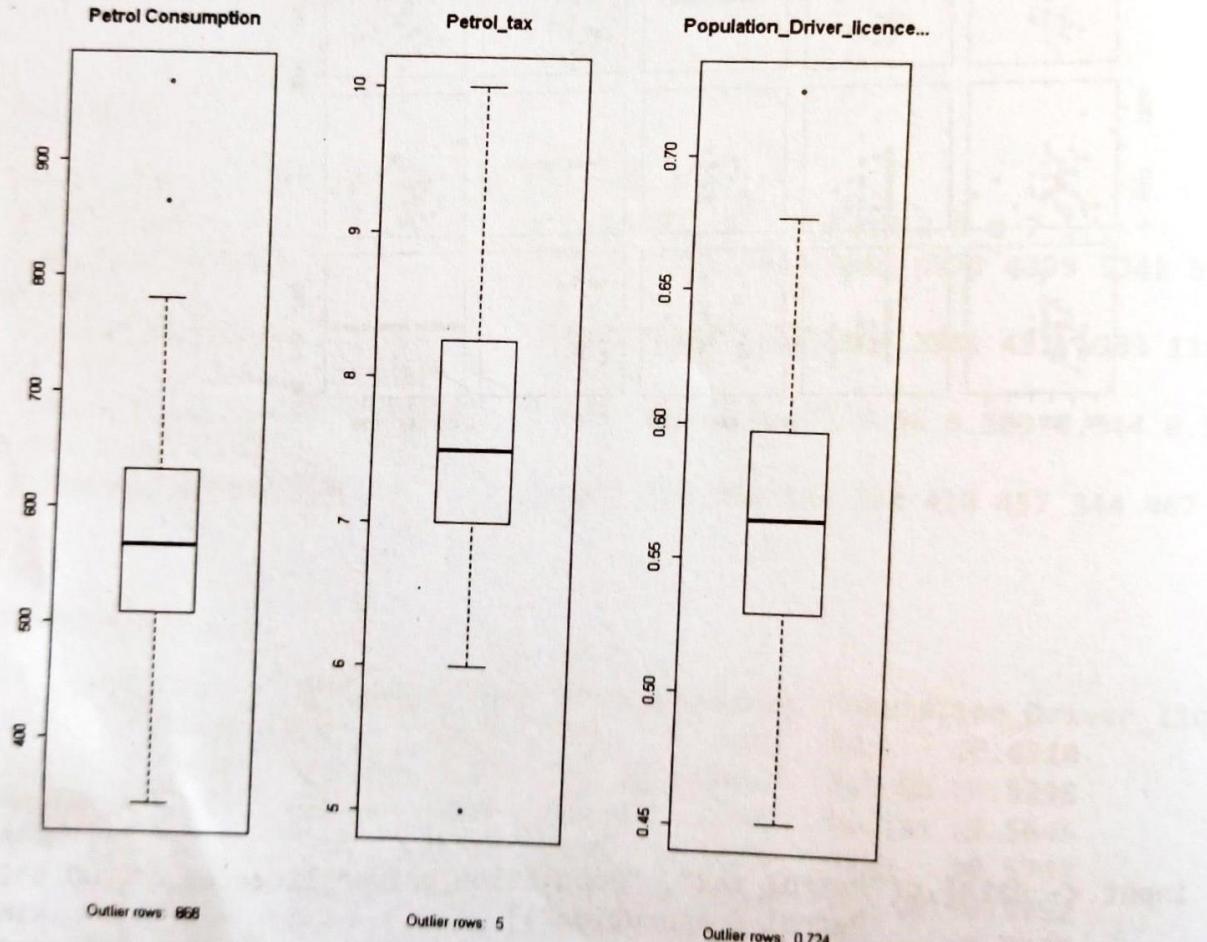
```
In [5]: # visualization
pairs(~Petrol_Consumption+Petrol_tax+Average_income+Paved_Highways+Population_Driver_licence..., data = data)
```



```
In [6]: input <- data[,c("Petrol_tax", "Population_Driver_licence...", "Petrol_Consumption")]
print(head(input))
```

	Petrol_tax	Population_Driver_licence...	Petrol_Consumption
1	9.0	0.525	541
2	9.0	0.572	524
3	9.0	0.580	561
4	7.5	0.529	414
5	8.0	0.544	410
6	10.0	0.571	457

```
In [7]: # BoxPlot to Check for outliers
par(mfrow=c(1, 3)) # divide graph area in 3 columns
boxplot(input$Petrol_Consumption,
        main="Petrol Consumption",
        sub=paste("Outlier rows: ",
                  boxplot.stats(input$Petrol_Consumption)$out))
boxplot(input$Petrol_tax,
        main="Petrol_tax",
        sub=paste("Outlier rows: ",
                  boxplot.stats(input$Petrol_tax)$out))
boxplot(input$Population_Driver_licence...,
        main="Population_Driver_licence...",
        sub=paste("Outlier rows: ",
                  boxplot.stats(input$Population_Driver_licence...)$out))
```





```
In [8]: pc_outliers <- boxplot.stats(input$Petrol_Consumption)$out
print("pc_outliers: ")
print(pc_outliers)

pt_outliers <- c(boxplot.stats(input$Petrol_tax)$out)
print("pt_outliers: ")
print(pt_outliers)

pd_outliers <- c(boxplot.stats(input$Population_Driver_licence...)$out)
print("pd_outliers: ")
print(pd_outliers)

[1] "pc_outliers: "
[1] 865 968
[1] "pt_outliers: "
[1] 5
[1] "pd_outliers: "
[1] 0.724
```

```
In [9]: #drop rows have outliers
print(paste("Before drop:", nrow(input)))

for (record in pc_outliers){
  input <- input[input$Petrol_Consumption != record, ]
}
for (record in pt_outliers)
{
  input <- input[input$Petrol_tax != record, ]
}
for (record in pd_outliers)
{
  input <- input[input$Population_Driver_licence... != record, ]
}

print(paste("After drop:", nrow(input)))

[1] "Before drop: 48"
[1] "After drop: 45"
```

```
In [10]: # calculate correlation between
print("Correlations pc vs pt and pdl:")
print(cor(input$Petrol_Consumption,
          input$Petrol_tax))
print(cor(input$Petrol_Consumption,
          input$Population_Driver_licence...))

[1] "Correlations pc vs pt and pdl:"
[1] -0.4629515
[1] 0.6052256
```

In [11]: # Create the training (development) and test (validation) data.

```
set.seed(42) # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(input), 0.8*nrow(input))
print("Selected training row indexes:")
print(trainingRowIndex)
trainingData <- input[trainingRowIndex, ] # training data
testData <- input[-trainingRowIndex, ] # test data
print("Rows of training data and test data:")
print(nrow(trainingData))
print(nrow(testData))

[1] "Selected training row indexes:"
[1] 42 45 13 35 27 21 29 6 25 26 17 37 31 9 15 39 30 4 43 33 23 28 34 40 2
[26] 11 8 44 19 14 12 36 38 32 1 16
[1] "Rows of training data and test data:"
[1] 36
[1] 9
```

In [12]: # Create the relationship model.

```
lmMod <- lm(Petrol_Consumption~Petrol_tax+Population_Driver_licence...,  
            data = trainingData)
```

In [13]: cPred <- predict(lmMod, testData) # predict Petrol Consumption

```
# mean square error according to model
mse <- mean(lmMod$residuals^2)
print(paste("mse: ", mse))

# mean square error of testData
mse_test = mean((testData$Petrol_Consumption - cPred)^2)
print(paste("mse in test: ", mse_test))

[1] "mse: 3188.42645742408"
[1] "mse in test: 7706.63929472164"
```

In [14]: # Show the model.
print(summary(lmMod))

Call:

lm(formula = Petrol_Consumption ~ Petrol_tax + Population_Driver_licence...,
data = trainingData)

Residuals:

Min	1Q	Median	3Q	Max
-128.93	-50.19	9.01	43.28	114.56

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	311.51	175.40	1.776	0.084953 .
Petrol_tax	-27.13	11.65	-2.328	0.026180 *
Population_Driver_licence...	822.05	220.98	3.720	0.000739 ***

Signif. codes:	0 **** 0.001 ** 0.01 * 0.05 . 0.1 ' ' 1			

Residual standard error: 58.98 on 33 degrees of freedom

Multiple R-squared: 0.4469, Adjusted R-squared: 0.4134

F-statistic: 13.33 on 2 and 33 DF, p-value: 5.698e-05

In [15]: # => r^2 has Low value, this model fits ~ 45% data => not good!

In [16]: # Get the Intercept and coefficients as vector elements.

cat("# # # # The Coefficient Values # # # ", "\n")

b <- coef(lmMod)[1]
print(b)

mph <- coef(lmMod)[2]
mpd <- coef(lmMod)[3]

print(mph)
print(mpd)

The Coefficient Values # #

(Intercept)

311.5122

Petrol_tax

-27.12654

Population_Driver_licence...

822.047

In [17]: # new predictions
#pt = 9, pd = 0.58
x1 <- 9
x2 <- 0.58


```
y <- (mph*x1 + mpd*x2 + b)
print("Solution 1 - results:")
print(y)

# solution 2
y1 <- predict(lmMod, data.frame(Petrol_tax = x1,
                                  Population_Driver_licence... = x2))
print("Solution 2 - results:")
print(y1)
```



```
[1] "Solution 1 - results:"
Petrol_tax
544.1606
[1] "Solution 2 - results:"
1
544.1606
```

Yêu cầu 2: Áp dụng BMA cho dữ liệu trên để lựa chọn model với các thuộc tính phù hợp cho việc dùng Linear Regression dự đoán Petrol_Consumption

In [18]: library(BMA)


```
Loading required package: survival
Loading required package: leaps
Loading required package: robustbase
```



```
Attaching package: 'robustbase'

The following object is masked from 'package:survival':
heart
```



```
Loading required package: inline
Loading required package: rrcov
```


Scalable Robust Estimators with High Breakdown Point (version 1.4-3)

In [19]:
yvar = data[, ("Petrol_Consumption")]
xvars = data[, c(-5)]
bma = bicreg(xvars, yvar, strict = F, OR=2)
summary(bma)
imageplot.bma(bma)

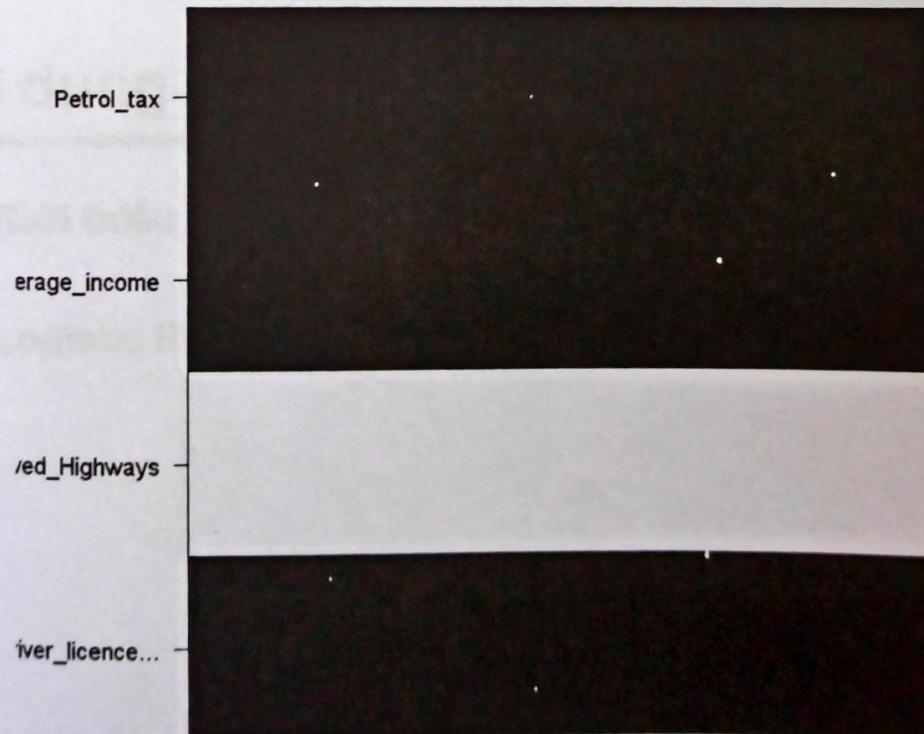
Call:
bicreg(x = xvars, y = yvar, strict = F, OR = 2)

1 models were selected
Best 1 models (cumulative posterior probability = 1):

	p!=0	EV	SD	model 1
Intercept	100	307.32790	156.83067	307.32790
Petrol_tax	100	-29.48381	10.58358	-29.48381
Average_income	100	-0.06802	0.01701	-0.06802
Paved_Highways	0	0.00000	0.00000	.
Population_Driver_licence...	100	1374.76841	183.66954	1374.76841

nVar	3
r2	0.675
BIC	-42.31437
post prob	1

Models selected by BMA



3/4/2020

Chapter17_ex2 - Jupyter Notebook

In [20]: # Select model with: Petrol_tax, Average_income, Population_Driver_Licence...