



## R for Data Science

### Bài 18: *Logistic 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. Logistic Regression



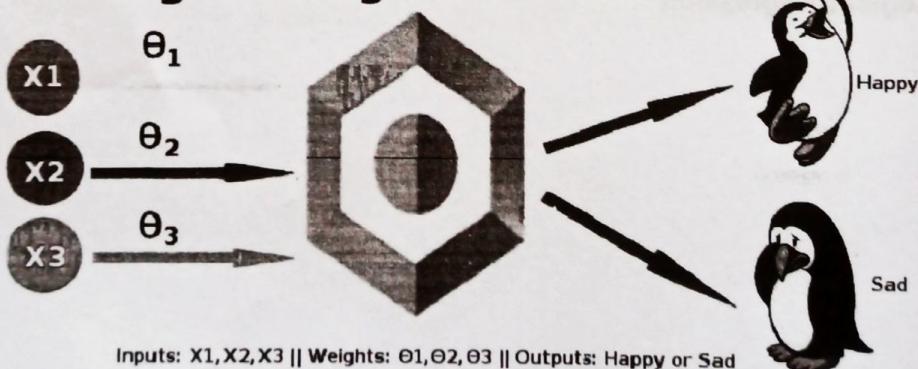
## Giới thiệu

- ❑ Logistic Regression là một thuật toán thuộc nhóm Supervised Learning sử dụng rất hiệu quả cho classification.
- ❑ Logistic Regression (còn gọi là Logit Regression) thường được sử dụng để ước tính xác suất mà một mẫu thuộc về một lớp cụ thể (ví dụ: xác suất một email có phải là spam hay không?).



## Giới thiệu

### Logistic Regression Model



## Giới thiệu

- Nếu xác suất ước tính lớn hơn 50%, thì mô hình dự đoán mẫu thuộc về lớp đó (được gọi là lớp positive, có nhãn "1"), hoặc ngược lại dự đoán rằng nó không thuộc về lớp đó (được gọi là lớp negative, có nhãn "0") => tạo ra một phân loại nhị phân, Binary Classifier**
  
- Binary Classifier (phân loại nhị phân): outcome chỉ có 1 và 0 hay đúng và sai dù trong tên có Regression, có thể gọi là Conditional Class probabilities.**

R programming language for Data Science

5

## Các ứng dụng

Credit Card Fraud (Gian lận thẻ tín dụng): Dự đoán liệu một giao dịch thẻ tín dụng có gian lận hay không?



Spam Detection :  
dự đoán 1 email có là Spam hay không?

## Các ứng dụng

- Health (Y tế):** Dự đoán một khái mô lanh tính hoặc ác tính?
- Marketing (Tiếp thị):** Dự đoán liệu một người dùng cụ thể có mua sản phẩm bảo hiểm hay không?
- Banking (Ngân hàng):** Dự đoán liệu khách hàng sẽ mặc định cho một khoản vay không?



## Ưu/ khuyết điểm

### Ưu điểm

- Không đưa ra giả định về phân phối các lớp trong không gian đặc trưng
- Dễ dàng mở rộng đến nhiều lớp (đa thức hồi quy)
- Huấn luyện nhanh
- Rất nhanh khi phân loại các bản ghi không xác định
- Độ chính xác cao cho nhiều tập dữ liệu đơn giản
- Khả năng phản ứng với overfitting
- Có thể giải thích các hệ số mô hình như các chỉ số về tầm quan trọng của tính năng

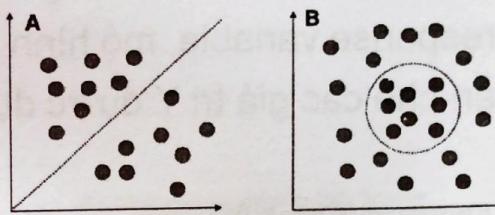


## Ưu/ khuyết điểm

### Khuyết điểm

- Ranh giới quyết định tuyến tính: Mô hình này chỉ phù hợp với loại dữ liệu mà hai class là gần với linearly separable. Một kiểu dữ liệu mà Logistic Regression không làm việc được là khi dữ liệu mà một class chứa các điểm nằm trong 1 vòng tròn, class kia chứa các điểm bên ngoài đường tròn đó.

Linear vs. nonlinear problems



9

## Lý do chọn

### Khuyết điểm

- Một hạn chế nữa của Logistic Regression là nó yêu cầu các điểm dữ liệu được tạo ra một cách độc lập với nhau. Trên thực tế, các điểm dữ liệu có thể bị ảnh hưởng bởi nhau.



## Lý do chọn

### Lý do không áp dụng Linear Regression với bài toán chỉ có hai outcome

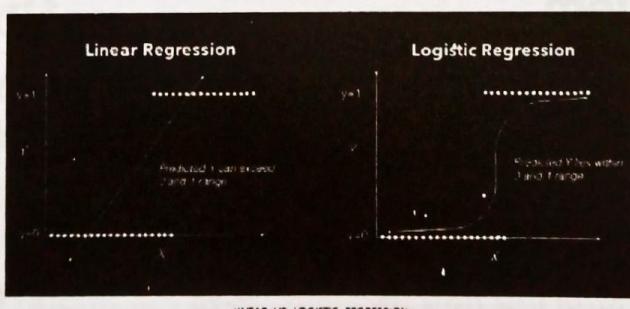
- Khi response variable chỉ có hai giá trị cụ thể, ta mong muốn có một mô hình dự đoán giá trị là 0 hoặc 1 hoặc là một điểm xác suất nằm trong khoảng từ 0 đến 1. Linear Regression không có khả năng này. Do đó, nếu ta sử dụng nó để mô hình binary response variable, mô hình kết quả có thể không hạn chế các giá trị Y được dự đoán trong 0 và 1



R programming language for Data Science

11

## Lý do chọn



Y được dự đoán có thể vượt quá khoảng 0..1

Y được dự đoán nằm trong khoảng 0..1

Trong logistic regression, ta nhận được một probability score, điểm xác suất, phản ánh xác suất xảy ra sự kiện.



R programming language for Data Science

12

# Nội dung

1. Giới thiệu
2. Logistic Regression



## Logistic Regression

**Là một mô hình hồi quy**, trong đó response variable (biến phụ thuộc) có các giá trị phân loại như TRUE/FALSE hoặc 1/0. Nó đo lường xác suất của một phản ứng nhị phân như giá trị của response variable dựa trên phương trình toán học liên quan tới response variable và các biến predictor variable.



## Logistic Regression

### □ Phương trình toán học

$$y = 1/(1+e^{-(a+b_1*x_1+b_2*x_2+b_3*x_3+\dots)})$$

### □ Trong đó:

- y: response variable
- $x_1, x_2, \dots, x_n$ : các predictor variable
- a,  $b_1, b_2, \dots, b_n$  là những hằng số được gọi là hệ số (coefficient).



## Logistic 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 `glm(formula, trainData, family)` với `trainData`, `family` là `binomial`
- In summary của model
- Sử dụng mô hình để dự đoán cho test data dùng `predict(mylogit, newdata = testData, type = "response")`
- Tính toán độ chính xác của model
- Dự đoán kết quả cho các sample mới theo `predict()`



# Logistic Regression

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

```
library(corrplot)
mydata <- read.csv("~/R/Chapter_Logistic_Regression/binary.csv")

## view the first few rows of the data
print(head(mydata))
print(tail(mydata))
print(summary(mydata))

  admit gre gpa rank
1     0 380 3.61   3
2     1 660 3.67   3
3     1 800 4.00   1
4     1 640 3.19   4
5     0 520 2.93   4
6     1 760 3.00   2

  admit gre gpa rank
395     1 460 3.99   3
396     0 620 4.00   2
397     0 560 3.04   3
398     0 460 2.63   2
399     0 700 3.65   2
400     0 600 3.89   3

  admit      gre      gpa      rank
Min. :0.0000  Min. :220.0  Min. :2.260  Min. :1.000
1st Qu.:0.0000  1st Qu.:520.0  1st Qu.:3.130  1st Qu.:2.000
Median :0.0000  Median :580.0  Median :3.395  Median :2.000
Mean   :0.3175  Mean   :587.7  Mean   :3.390  Mean   :2.485
3rd Qu.:1.0000  3rd Qu.:660.0  3rd Qu.:3.670  3rd Qu.:3.000
Max.  :1.0000  Max.  :800.0  Max.  :4.000  Max.  :4.000
```

R programming language for Data Science

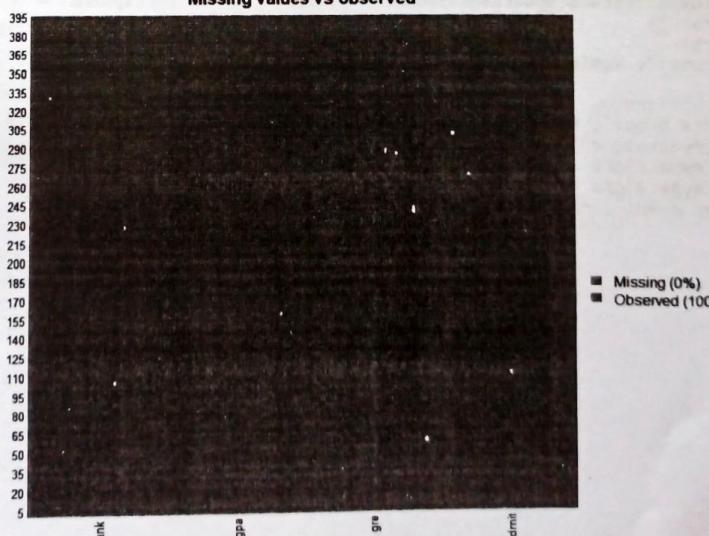
17



# Logistic Regression

```
# check missing value
library(Amelia)
missmap(mydata, main = "Missing values vs observed")
```

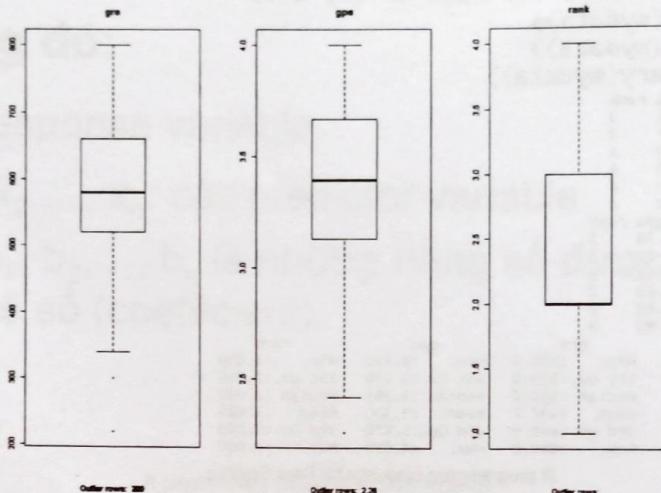
Missing values vs observed



18

# Logistic Regression

```
# BoxPlot to Check for outliers
par(mfrow=c(1, 3)) # divide graph area in 3 columns
boxplot(mydata$gre, main="gre",
        sub=paste("outlier rows: ", boxplot.stats(mydata$gre)$out)) # box plot for 'gre'
boxplot(mydata$gpa, main="gpa",
        sub=paste("outlier rows: ", boxplot.stats(mydata$gpa)$out)) # box plot for 'gpa'
boxplot(mydata$rank, main="rank",
        sub=paste("outlier rows: ", boxplot.stats(mydata$rank)$out)) # box plot for 'rank'
```



19



# Logistic Regression

```
gre_outliers <- boxplot.stats(mydata$gre)$out
print("gre_outliers: ")
print(gre_outliers)
print(paste("Numrows: ", sum(mydata$gre == gre_outliers[1]) + sum(mydata$gre == gre_outliers[3])))

gpa_outliers <- boxplot.stats(mydata$gpa)$out
print("wt_outliers: ")
print(gpa_outliers)
print(paste("Numrows: ", sum(mydata$gpa == gpa_outliers[1])))

#drop rows have outliers
print(paste("Before drop: ", nrow(mydata)))
mydata <- mydata[mydata$gre != gre_outliers[1],]
mydata <- mydata[mydata$gre != gre_outliers[3],]
mydata <- mydata[mydata$gpa != gpa_outliers[1],]
print(paste("After drop: ", nrow(mydata)))
```

```
[1] "gre_outliers: "
[1] 300 300 220 300
[1] "Numrows: 4"
[1] "wt_outliers: "
[1] 2.26
[1] "Numrows: 1"
[1] "Before drop: 400"
[1] "After drop: 395"
```

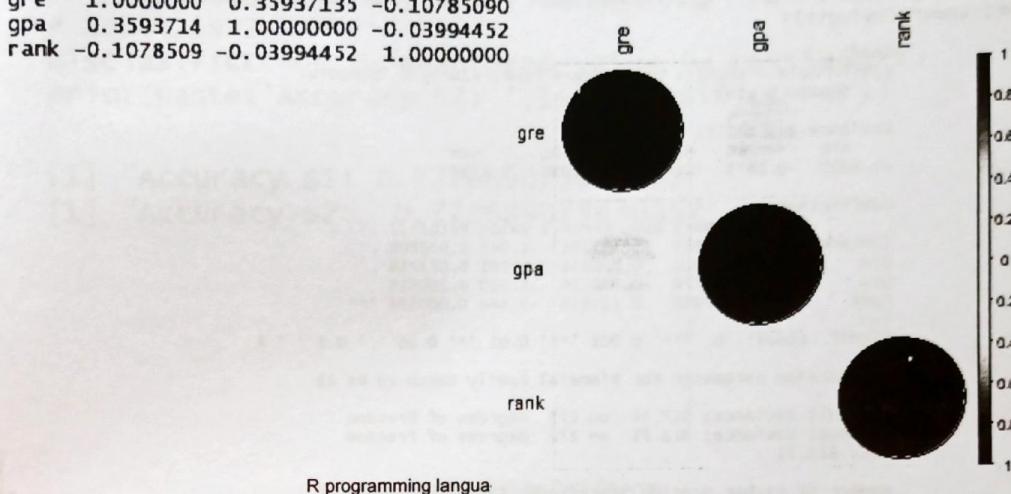


# Logistic Regression



```
# calculating the correlation between each pair of numeric variables
correlations <- cor(mydata[,2:4])
print(correlations)
corrplot(correlations, method="circle")
```

	gre	gpa	rank
gre	1.0000000	0.35937135	-0.10785090
gpa	0.3593714	1.00000000	-0.03994452
rank	-0.1078509	-0.03994452	1.00000000



# Logistic Regression



- Tạo training data và test data

```
# divided into train and test: 70 - 30
n = nrow(mydata)
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
train = mydata[trainIndex ,]
test = mydata[-trainIndex ,]
print("Rows of training data and test data:")
print(nrow(train))
print(nrow(test))
```

```
[1] "Rows of training data and test data:"
[1] 276
[1] 119
```



# Logistic Regression

- Xây dựng mô hình (model) sử dụng glm(formula, trainData, family) với training data, family là binomial

## In summary của model

```
# estimates a logistic regression model using the glm (generalized linear model) function
mylogit <- glm(admit ~ gre + gpa + rank, data = train, family = "binomial")
print(summary(mylogit))

Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
     data = train)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.4600 -0.8675 -0.6435  1.2058  2.1734 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -2.681833  1.381381 -1.941 0.052208 .  
gre          0.002420  0.001351  1.791 0.073218 .  
gpa          0.482778  0.377229  1.280 0.200616    
rank         -0.527093  0.153058 -3.444 0.000574 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 337.54  on 275  degrees of freedom
Residual deviance: 315.71  on 272  degrees of freedom
AIC: 323.71 → cảng nhỏ cảng bờ

Number of Fisher Scoring iterations: 4
```

23

# Logistic Regression

- Sử dụng mô hình để dự đoán cho test data dùng predict(mylogit, newdata = testData, type = "response")

```
pred = predict(mylogit,
               newdata = test,
               type = "response")

pred_value <- ifelse(pred > 0.5, 1, 0)
print("Testdata admit vs predict (10 rows:")
result <- data.frame(testAdmit = test$admit[30:40], pred_value[30:40])
print(result)

[1] "Testdata admit vs predict (10 rows:"
testAdmit pred_value
  103        0        0
  107        1        1
  110        0        0
  111        0        0
  117        1        0
  121        1        0
  129        0        0
  130        0        0
  136        0        0
  138        0        0
  142        1        0
```

# Logistic Regression



- Tính toán độ chính xác của model

```
# SOLUTION 1  
accuracy <- table(pred_value, test[, "admit"])  
accuracy = sum(diag(accuracy))/sum(accuracy)  
print(paste("Accuracy s1:", accuracy))  
# SOLUTION 2  
misClasificError <- mean(pred_value != test$admit)  
print(paste('Accuracy s2: ', 1-misClasificError))
```

```
[1] "Accuracy s1: 0.722689075630252"  
[1] "Accuracy s2: 0.722689075630252"
```



# Logistic Regression



- Dự đoán kết quả cho các sample mới theo predict()

```
# make new prediction  
y1 <- predict(mylogit,  
               newdata = data.frame(gre = c(580, 800),  
                                         gpa = c(3.4, 4),  
                                         rank = c(3, 1)),  
               type='response')  
y1 <- ifelse(y1 > 0.5, 1, 0)  
print("results:")  
print(y1)  
  
[1] "results:"  
1 2  
0 1
```



# Chapter 18: Logistic Regression

## Exercise 1: Diabetes

Cho dữ liệu diabetes.csv

Thông tin các cột dữ liệu:

- Pregnancies: số lần mang thai
- Glucose: Nồng độ glucose huyết tương 2 giờ trong thử nghiệm dung nạp glucose đường uống
- BloodPressure: Huyết áp tâm trương (mm Hg)
- SkinThickness: độ dày da gấp Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml). insulin huyết thanh 2-giờ
- BMI: (weight in kg/(height in m)<sup>2</sup>)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

Chú ý: Tất cả các biến trên liên tục, mục đích là dự đoán ai đó có bị tiểu đường hay không (Outcome=1) dựa trên các biến khác. Các mẫu lấy từ phụ nữ trên 21 years old.

**Yêu cầu: Áp dụng LogisticRegression để thực hiện việc dự đoán khả năng dương tính với bệnh tiểu đường (positive diabete - outputs) dựa trên các biến lâm sàng khác (clinical variables - inputs)**

- Hãy áp dụng LogisticRegression để dự đoán khả năng dương tính với bệnh tiểu đường
- 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 các biến (corrplot)
- Tạo train:test từ dữ liệu data với tỉ lệ 70:30
- Áp dụng thuật toán LogisticRegression
- Tìm kết quả
- Trực quan hóa kết quả
- Hãy cho biết với những người có pregnant, glucose, pressure, triceps, insulin, mass, pedigree, age lần lượt như sau thì ai có khả năng dương tính với bệnh tiểu đường, ai không?
  - 8, 176, 90, 34, 300, 33.7, 0.467, 58
  - 1, 100, 66, 15, 56, 23.6, 0.666, 26
  - 12, 88, 74, 40, 54, 35.3, 0.378, 48

```
In [1]: library(corrplot)
mydata <- read.csv("diabetes.csv")
```

```
In [2]: ## view the first few rows of the data  
print(head(mydata))
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
1	6	148	72	35	0	33.6
2	1	85	66	29	0	26.6
3	8	183	64	0	0	23.3
4	1	89	66	23	94	28.1
5	0	137	40	35	168	43.1
6	5	116	74	0	0	25.6

	DiabetesPedigreeFunction	Age	Outcome
1	0.627	50	1
2	0.351	31	0
3	0.672	32	1
4	0.167	21	0
5	2.288	33	1
6	0.201	30	0

```
In [3]: # print(tail(mydata))
```

```
In [4]: print(summary(mydata))
```

Pregnancies	Glucose	BloodPressure	SkinThickness
Min. : 0.000	Min. : 0.0	Min. : 0.00	Min. : 0.00
1st Qu.: 1.000	1st Qu.: 99.0	1st Qu.: 62.00	1st Qu.: 0.00
Median : 3.000	Median :117.0	Median : 72.00	Median :23.00
Mean : 3.845	Mean :120.9	Mean : 69.11	Mean :20.54
3rd Qu.: 6.000	3rd Qu.:140.2	3rd Qu.: 80.00	3rd Qu.:32.00
Max. :17.000	Max. :199.0	Max. :122.00	Max. :99.00

Insulin	BMI	DiabetesPedigreeFunction	Age
Min. : 0.0	Min. : 0.00	Min. :0.0780	Min. :21.00
1st Qu.: 0.0	1st Qu.:27.30	1st Qu.:0.2437	1st Qu.:24.00
Median : 30.5	Median :32.00	Median :0.3725	Median :29.00
Mean : 79.8	Mean :31.99	Mean :0.4719	Mean :33.24
3rd Qu.:127.2	3rd Qu.:36.60	3rd Qu.:0.6262	3rd Qu.:41.00
Max. :846.0	Max. :67.10	Max. :2.4200	Max. :81.00

Outcome
Min. :0.000
1st Qu.:0.000
Median :0.000
Mean :0.349
3rd Qu.:1.000
Max. :1.000

In [5]: `print(str(mydata))`

```
'data.frame': 768 obs. of 9 variables:  
 $ Pregnancies           : int 6 1 8 1 0 5 3 10 2 8 ...  
 $ Glucose                : int 148 85 183 89 137 116 78 115 197 125 ...  
 $ BloodPressure          : int 72 66 64 66 40 74 50 0 70 96 ...  
 $ SkinThickness          : int 35 29 0 23 35 0 32 0 45 0 ...  
 $ Insulin                : int 0 0 0 94 168 0 88 0 543 0 ...  
 $ BMI                    : num 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0  
 ...  
 $ DiabetesPedigreeFunction: num 0.627 0.351 0.672 0.167 2.288 ...  
 $ Age                    : int 50 31 32 21 33 30 26 29 53 54 ...  
 $ Outcome                : int 1 0 1 0 1 0 1 0 1 1 ...  
NULL
```

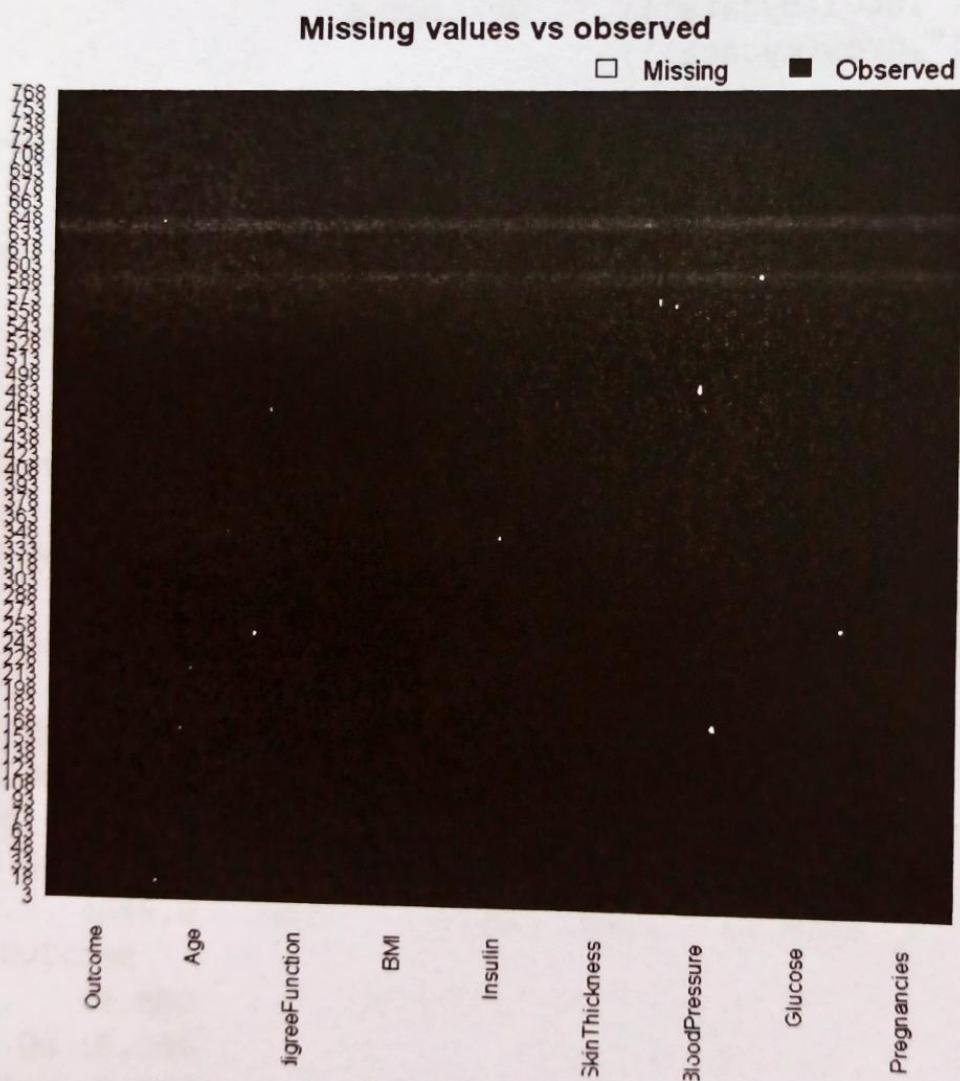
In [6]: `print(paste("cols:", ncol(mydata)))`  
`print(paste("rows:", nrow(mydata)))`

```
[1] "cols: 9"  
[1] "rows: 768"
```

```
In [7]: # check missing value
library(Amelia)
missmap(mydata, main = "Missing values vs observed")
# => no missing values
```

Loading required package: Rcpp

```
##
## Amelia II: Multiple Imputation
## (Version 1.7.4, built: 2015-12-05)
## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ (http://gking.harvard.edu/amelia/)
for more information
##
```



```
In [8]: # Check Class bias  
print(table(mydata$Outcome))
```

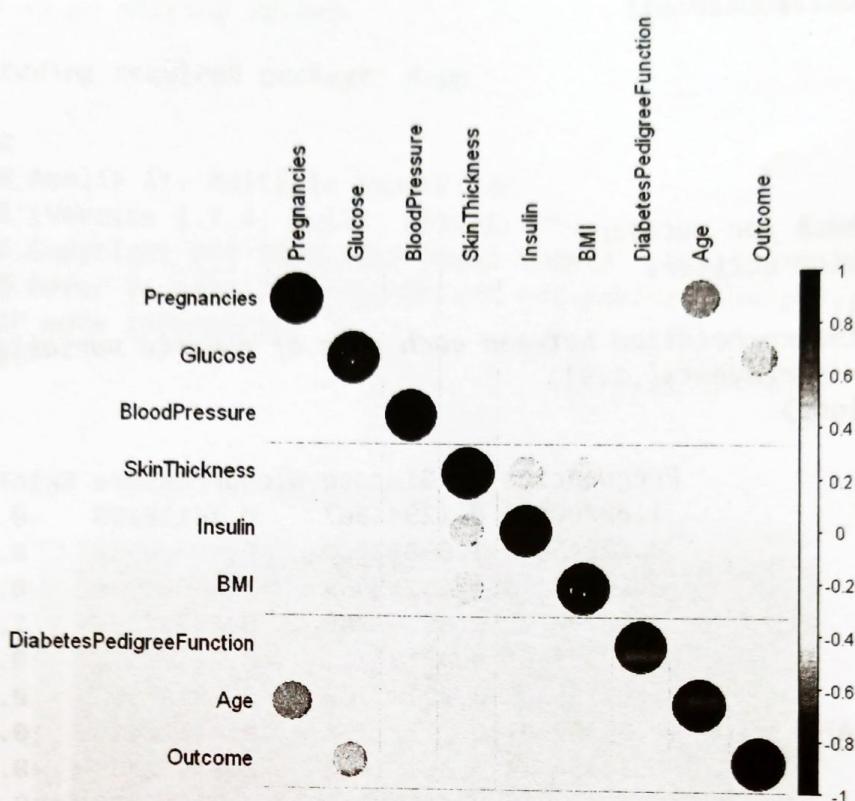
0 1  
500 268

```
In [9]: # BoxPlot to Check for outliers  
# Drop rows having outliers
```

# calculating the correlation between each pair of numeric variables  
correlations <- cor(mydata[,1:9])  
print(correlations)

	Pregnancies	Glucose	BloodPressure	SkinThickness
Pregnancies	1.0000000	0.12945867	0.14128198	-0.08167177
Glucose	0.12945867	1.0000000	0.15258959	0.05732789
BloodPressure	0.14128198	0.15258959	1.0000000	0.20737054
SkinThickness	-0.08167177	0.05732789	0.20737054	1.0000000
Insulin	-0.07353461	0.33135711	0.08893338	0.43678257
BMI	0.01768309	0.22107107	0.28180529	0.39257320
DiabetesPedigreeFunction	-0.0352267	0.13733730	0.04126495	0.18392757
Age	0.54434123	0.26351432	0.23952795	-0.11397026
Outcome	0.22189815	0.46658140	0.06506836	0.07475223
	Insulin	BMI	DiabetesPedigreeFunction	
Pregnancies	-0.07353461	0.01768309	-0.03352267	
Glucose	0.33135711	0.22107107	0.13733730	
BloodPressure	0.08893338	0.28180529	0.04126495	
SkinThickness	0.43678257	0.39257320	0.18392757	
Insulin	1.0000000	0.19785906	0.18507093	
BMI	0.19785906	1.0000000	0.14064695	
DiabetesPedigreeFunction	0.18507093	0.14064695	1.0000000	
Age	-0.04216295	0.03624187	0.03356131	
Outcome	0.13054795	0.29269466	0.17384407	
	Age	Outcome		
Pregnancies	0.54434123	0.22189815		
Glucose	0.26351432	0.46658140		
BloodPressure	0.23952795	0.06506836		
SkinThickness	-0.11397026	0.07475223		
Insulin	-0.04216295	0.13054795		
BMI	0.03624187	0.29269466		
DiabetesPedigreeFunction	0.03356131	0.17384407		
Age	1.0000000	0.23835598		
Outcome	0.23835598	1.0000000		

```
In [10]: corrplot(correlations, method="circle")
```



```
In [11]: # divided into train and test: 70 - 30
```

```
n = nrow(mydata)
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
train = mydata[trainIndex ,]
test = mydata[-trainIndex ,]
print("Rows of training data and test data:")
print(nrow(train))
print(nrow(test))

[1] "Rows of training data and test data:"
[1] 538
[1] 230
```



```
In [12]: # estimates a logistic regression model using the glm (generalized linear
mylogit <- glm(Outcome ~ ., data = train, family = "binomial")
print(summary(mylogit))
```

Call:

```
glm(formula = Outcome ~ ., family = "binomial", data = train)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.5598	-0.7087	-0.3912	0.6966	3.0360

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-8.617615	0.868548	-9.922	< 2e-16 ***
Pregnancies	0.138184	0.040714	3.394	0.000689 ***
Glucose	0.038237	0.004636	8.247	< 2e-16 ***
BloodPressure	-0.017082	0.006894	-2.478	0.013212 *
SkinThickness	0.008086	0.008332	0.971	0.331773
Insulin	-0.001323	0.001086	-1.219	0.222855
BMI	0.089461	0.018447	4.850	1.24e-06 ***
DiabetesPedigreeFunction	0.837717	0.371315	2.256	0.024065 *
Age	0.012923	0.011409	1.133	0.257341

---

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 692.48 on 537 degrees of freedom

Residual deviance: 490.74 on 529 degrees of freedom

AIC: 508.74

Number of Fisher Scoring iterations: 5

```
In [13]: pred = predict(mylogit,
                      newdata = test,
                      type = "response")

pred_value <- ifelse(pred > 0.5, 1, 0)
print("Testdata admit vs predict (10 rows:")
result <- data.frame(testAdmit = test$Outcome[30:40], pred_value[30:40])
print(result)

[1] "Testdata admit vs predict (10 rows:"
  testAdmit pred_value.30.40.
  120      0      0
  121      1      1
  123      0      0
  127      0      0
  129      1      0
  132      1      1
  134      0      0
  135      0      0
  136      0      0
  140      0      0
  143      0      0
```

```
In [14]: print(pred_value[30:40])
print(test$Outcome[30:40])

120 121 123 127 129 132 134 135 136 140 143
  0   1   0   0   0   1   0   0   0   0   0
[1] 0 1 0 0 1 1 0 0 0 0 0
```

```
In [15]: # SOLUTION 1
accuracy <- table(pred_value, test[, "Outcome"])
accuracy = sum(diag(accuracy))/sum(accuracy)
print(paste("Accuracy s1:", accuracy))

[1] "Accuracy s1: 0.752173913043478"
```

```
In [16]: # SOLUTION 2
misClasificError <- mean(pred_value != test$Outcome)
print(paste('Accuracy s2: ', 1-misClasificError))

[1] "Accuracy s2: 0.752173913043478"
```

In [17]: `summary(mylogit)`

Call:

`glm(formula = Outcome ~ ., family = "binomial", data = train)`

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.5598	-0.7087	-0.3912	0.6966	3.0360

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	-8.617615	0.868548	-9.922	< 2e-16	***
Pregnancies	0.138184	0.040714	3.394	0.000689	***
Glucose	0.038237	0.004636	8.247	< 2e-16	***
BloodPressure	-0.017082	0.006894	-2.478	0.013212	*
SkinThickness	0.008086	0.008332	0.971	0.331773	
Insulin	-0.001323	0.001086	-1.219	0.222855	
BMI	0.089461	0.018447	4.850	1.24e-06	***
DiabetesPedigreeFunction	0.837717	0.371315	2.256	0.024065	*
Age	0.012923	0.011409	1.133	0.257341	

---

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 692.48 on 537 degrees of freedom  
Residual deviance: 490.74 on 529 degrees of freedom  
AIC: 508.74

Number of Fisher Scoring iterations: 5

```
In [18]: # make new prediction
#. 8, 176, 90, 34, 300, 33.7, 0.467, 58
#. 1, 100, 66, 15, 56, 23.6, 0.666, 26
#. 12, 88, 74, 40, 54, 35.3, 0.378, 48

print(colnames(test))

y1 <- predict(mylogit,
               newdata = data.frame(Pregnancies = c(8, 1, 12),
                                     Glucose = c(176, 100, 88),
                                     BloodPressure = c(90, 66, 74),
                                     SkinThickness = c(34, 15, 40),
                                     Insulin = c(300, 56, 54),
                                     BMI = c(33.7, 23.6, 35.3),
                                     DiabetesPedigreeFunction = c(0.467, 0.666, 0.378),
                                     Age = c(58, 26, 48)),
               type='response')
y1 <- ifelse(y1 > 0.5, 1, 0)
print("results:")
print(y1)

[1] "Pregnancies"           "Glucose"
[3] "BloodPressure"          "SkinThickness"
[5] "Insulin"                 "BMI"
[7] "DiabetesPedigreeFunction" "Age"
[9] "Outcome"
[1] "results:"
1 2 3
1 0 0
```

```
In [ ]:
```

# Chapter 18: Logistic Regression

## Exercise 2: Low birth weight?

**Yêu cầu:** Logistic Regression để thực hiện việc xác định trẻ có thiếu cân hay không dựa vào thông tin còn lại.

- Cho dữ liệu birthweight\_reduced.csv
- Tạo dataset
- In thông tin head, tail, số dòng, số cột, str, summary
- Vẽ biểu đồ quan sát mối liên hệ giữa các biến (corrplot)
- Tạo train:test từ dữ liệu data với tỉ lệ 80:20
- Áp dụng thuật toán Logistic Regression
- Kiểm tra độ chính xác
- Tìm kết quả Cho dữ liệu Test: c(12, 18, 4.5, 35, 1, 41, 7, 65, 125, 37, 14, 25, 68, 1, 1)

```
In [1]: library(corrplot)
mydata <- read.csv("birthweight_reduced.csv")

In [2]: print(str(mydata))

'data.frame': 42 obs. of 17 variables:
 $ id           : int 1313 431 808 300 516 321 1363 575 822 1081 ...
 $ headcircumference: int 12 12 13 12 13 13 12 12 13 14 ...
 $ length       : int 17 19 19 18 18 19 19 19 19 21 ...
 $ Birthweight   : num 5.8 4.2 6.4 4.5 5.8 6.8 5.2 6.1 7.5 8 ...
 $ Gestation     : int 33 33 34 35 35 37 37 37 38 38 ...
 $ smoker        : int 0 1 0 1 1 0 1 1 0 0 ...
 $ motherage     : int 24 20 26 41 20 28 20 19 20 18 ...
 $ mnocig        : int 0 7 0 7 35 0 7 7 0 0 ...
 $ mheight       : int 58 63 65 65 67 62 64 65 62 67 ...
 $ mppwt         : int 99 109 140 125 125 118 104 132 103 109 ...
 $ fage          : int 26 20 25 37 23 39 20 20 22 20 ...
 $ fedyrs        : int 16 10 12 14 12 10 10 14 14 12 ...
 $ fnocig        : int 0 35 25 25 50 0 35 0 0 7 ...
 $ fheight       : int 66 71 69 68 73 67 73 72 70 67 ...
 $ lowbwt        : int 1 1 0 1 1 0 1 0 0 0 ...
 $ mage35        : int 0 0 0 1 0 0 0 0 0 0 ...
 $ LowBirthWeight: Factor w/ 2 levels "Low","Normal": 1 1 2 1 1 2 1 2 2 2 ...
```

NULL

In [3]: ## view the first few rows of the data  
 print(head(mydata))  
 #print(tail(mydata))

	id	headcircumference	length	Birthweight	Gestation	smoker	motherage	mnocig	
1	1313	12	17	5.8	33	0	24	0	
2	431	12	19	4.2	33	1	20	7	
3	808	13	19	6.4	34	0	26	0	
4	300	12	18	4.5	35	1	41	7	
5	516	13	18	5.8	35	1	20	35	
6	321	13	19	6.8	37	0	28	0	
	mheight	mppwt	fage	fedyrs	fnocig	fheight	lowbwt	mage35	LowBirthWeight
1	58	99	26	16	0	66	1	0	Low
2	63	109	20	10	35	71	1	0	Low
3	65	140	25	12	25	69	0	0	Normal
4	65	125	37	14	25	68	1	1	Low
5	67	125	23	12	50	73	1	0	Low
6	62	118	39	10	0	67	0	0	Normal

In [4]: print(summary(mydata))

	id	headcircumference	length	Birthweight	
Min. :	27.0	Min. :12.00	Min. :17.00	Min. : 4.200	
1st Qu.:	537.2	1st Qu.:13.00	1st Qu.:19.00	1st Qu.: 6.450	
Median :	821.0	Median :13.00	Median :20.00	Median : 7.250	
Mean :	894.1	Mean :13.26	Mean :19.93	Mean : 7.264	
3rd Qu.:	1269.5	3rd Qu.:14.00	3rd Qu.:21.00	3rd Qu.: 8.000	
Max. :	1764.0	Max. :15.00	Max. :22.00	Max. :10.600	
	Gestation	smoker	motherage	mnocig	
Min. :	33.00	Min. :0.0000	Min. :18.00	Min. : 0.000	
1st Qu.:	38.00	1st Qu.:0.0000	1st Qu.:20.25	1st Qu.: 0.000	
Median :	39.50	Median :1.0000	Median :24.00	Median : 4.500	
Mean :	39.19	Mean :0.5238	Mean :25.55	Mean : 9.429	
3rd Qu.:	41.00	3rd Qu.:1.0000	3rd Qu.:29.00	3rd Qu.:15.750	
Max. :	45.00	Max. :1.0000	Max. :41.00	Max. :50.000	
	mheight	mppwt	fage	fedyrs	fnocig
Min. :	58.0	Min. : 99.0	Min. :19.0	Min. :10.00	Min. : 0.00
1st Qu.:	63.0	1st Qu.:115.0	1st Qu.:23.0	1st Qu.:12.00	1st Qu.: 0.00
Median :	64.0	Median :125.0	Median :29.5	Median :14.00	Median :18.50
Mean :	64.4	Mean :125.9	Mean :28.9	Mean :13.67	Mean :17.19
3rd Qu.:	66.0	3rd Qu.:135.0	3rd Qu.:32.0	3rd Qu.:16.00	3rd Qu.:25.00
Max. :	71.0	Max. :170.0	Max. :46.0	Max. :16.00	Max. :50.00
	fheight	lowbwt	mage35	LowBirthWeight	
Min. :	66.00	Min. :0.0000	Min. :0.00000	Low : 6	
1st Qu.:	69.00	1st Qu.:0.0000	1st Qu.:0.00000	Normal:36	
Median :	71.00	Median :0.0000	Median :0.00000		
Mean :	70.76	Mean :0.1429	Mean :0.09524		
3rd Qu.:	72.00	3rd Qu.:0.0000	3rd Qu.:0.00000		
Max. :	78.00	Max. :1.0000	Max. :1.00000		

In [5]: print(paste("rows:", ncol(mydata)))  
 print(paste("cols:", nrow(mydata)))

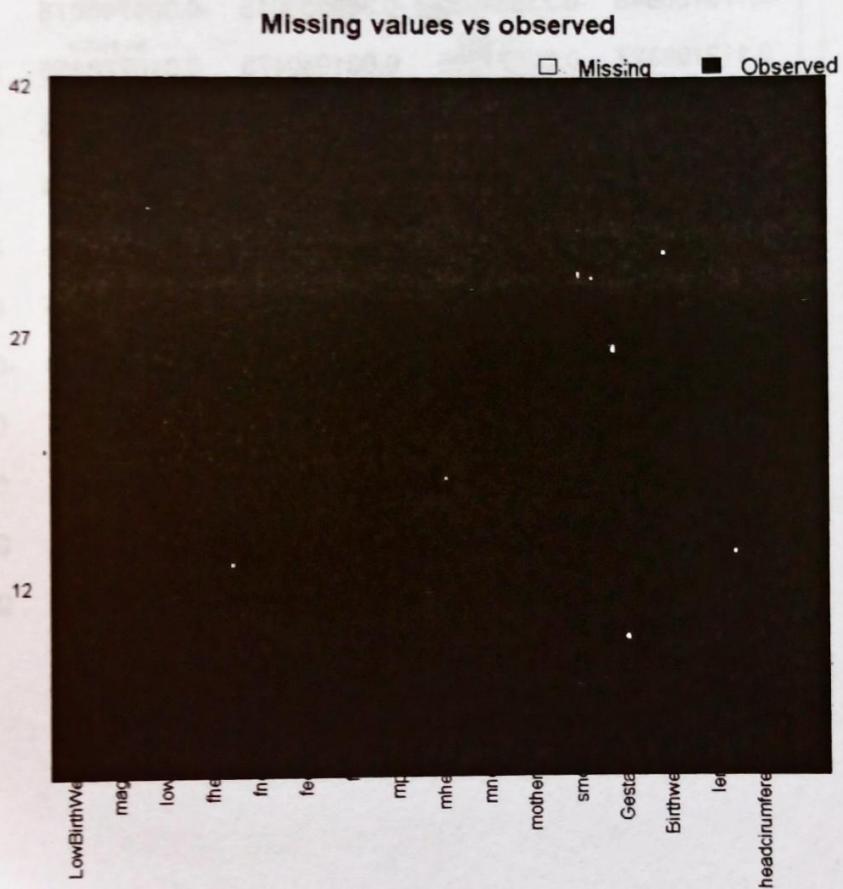
[1] "rows: 17"  
 [1] "cols: 42"



```
In [6]: # check missing value
library(Amelia)
missmap(mydata, main = "Missing values vs observed")

Loading required package: Rcpp

##
## Amelia II: Multiple Imputation
## (Version 1.7.4, built: 2015-12-05)
## Copyright (C) 2005-2020 James Honaker, Gary King and Matthew Blackwell
## Refer to http://gking.harvard.edu/amelia/ (http://gking.harvard.edu/amelia/)
for more information
##
```



```
In [7]: # Check Class bias
print(table(mydata$LowBirthWeight))
```

Low	Normal
6	36

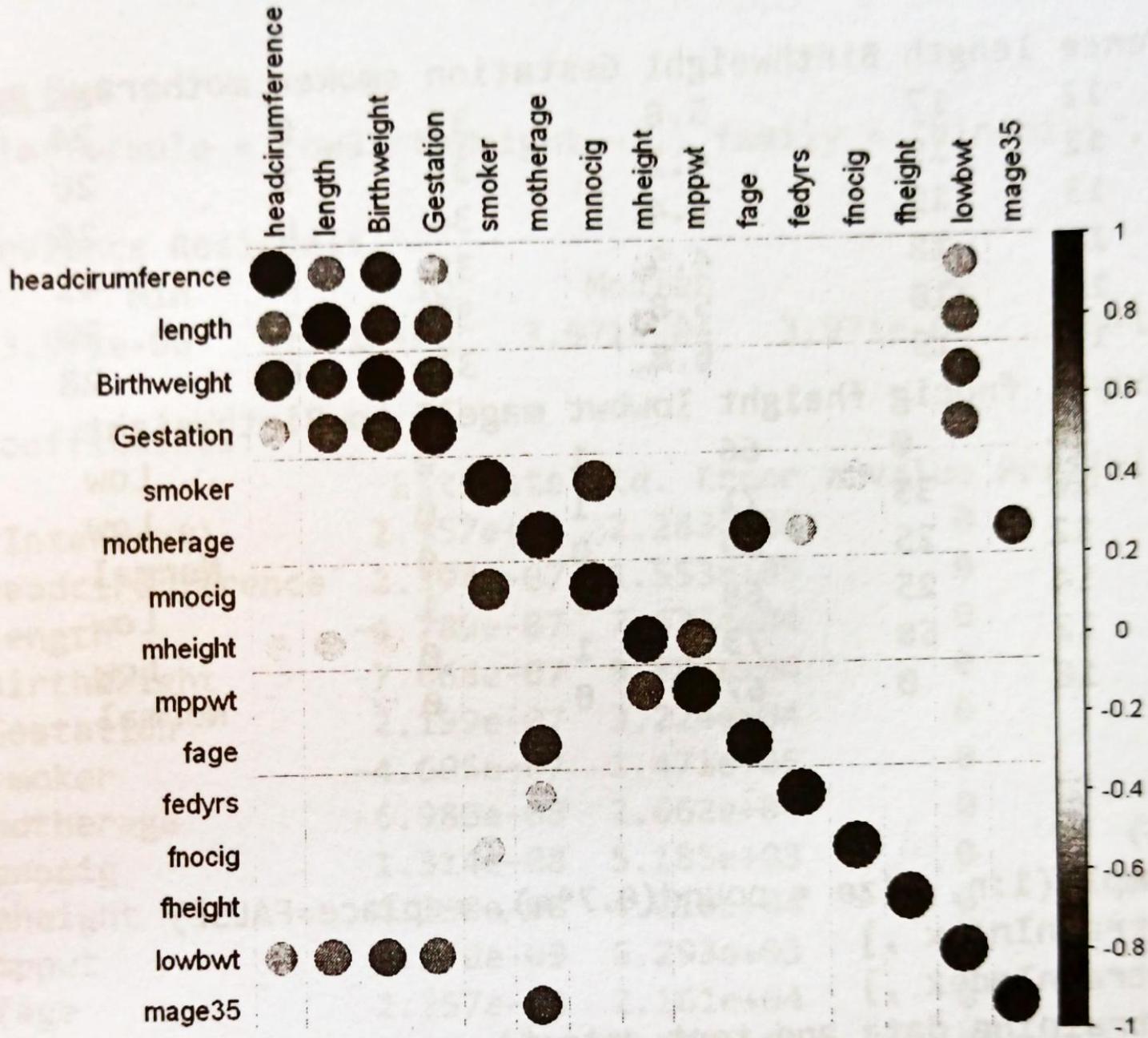
```
In [8]: # BoxPlot to Check for outliers
# drop rows having outliers

# calculating the correlation between each pair of numeric variables
correlations <- cor(mydata[,2:16])
correlations
```

	<b>headcircumference</b>	<b>length</b>	<b>Birthweight</b>	<b>Gestation</b>	<b>smoker</b>	<b>mothe</b>
<b>headcircumference</b>	1.000000000	0.56532849	0.736396310	0.443974538	-0.17375085	0.11210
<b>length</b>	0.565328491	1.00000000	0.697008279	0.651402769	-0.23534939	-0.02071
<b>Birthweight</b>	0.736396310	0.69700828	1.000000000	0.706291950	-0.30895001	0.00104
<b>Gestation</b>	0.443974538	0.65140277	0.706291950	1.000000000	-0.09474608	0.01077
<b>smoker</b>	-0.173750846	-0.23534939	-0.308950015	-0.094746078	1.00000000	0.21247
<b>motherage</b>	0.112108327	-0.02071895	0.001040475	0.010778455	0.21247879	1.00000
<b>mnocig</b>	-0.131437996	-0.15713803	-0.151227745	0.043194856	0.72721809	0.34029
<b>mheight</b>	0.381293418	0.41473145	0.367947042	0.230929298	0.03968201	0.04678
<b>mppwt</b>	0.357593509	0.30439408	0.389580646	0.250515534	0.01258798	0.27764
<b>fage</b>	0.301363456	0.07890718	0.176790000	0.142175334	0.19750145	0.80658
<b>fedyrs</b>	0.083416559	-0.05072288	0.073869580	0.130986636	-0.01489058	0.44168
<b>fnocig</b>	-0.027734282	0.01971581	-0.088927203	-0.113830614	0.41763296	0.09092
<b>fheight</b>	0.040466392	0.18713730	0.024784274	0.187866905	0.10583531	-0.20360
<b>lowbwt</b>	-0.500246731	-0.59224820	-0.651804466	-0.602934976	0.25301216	-0.07639
<b>mage35</b>	-0.005096869	0.02107483	-0.108480485	0.007394508	0.14693845	0.69266

2020

```
In [9]: corrplot(correlations, method="circle")
```





In [10]: # divided into train and test: 70 - 30

```
mydata <- mydata[, 2:17]
print(head(mydata))
```

	headcircumference	length	Birthweight	Gestation	smoker	motherage	mnocig	mheight
1	12	17	5.8	33	0	24	0	58
2	12	19	4.2	33	1	20	7	63
3	13	19	6.4	34	0	26	0	65
4	12	18	4.5	35	1	41	7	65
5	13	18	5.8	35	1	20	35	67
6	13	19	6.8	37	0	28	0	62
	mppwt	fage	fedyrs	fnocig	fheight	lowbwt	mage35	LowBirthWeight
1	99	26	16	0	66	1	0	Low
2	109	20	10	35	71	1	0	Low
3	140	25	12	25	69	0	0	Normal
4	125	37	14	25	68	1	1	Low
5	125	23	12	50	73	1	0	Low
6	118	39	10	0	67	0	0	Normal

In [11]: n = nrow(mydata)

```
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
train = mydata[trainIndex ,]
test = mydata[-trainIndex ,]
print("Rows of training data and test data:")
print(nrow(train))
print(nrow(test))
```

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

[1] 29

[1] 13



In [12]: # estimates a Logistic regression model using the glm (generalized Linear  
 mylogit <- glm(LowBirthWeight ~ ., data = train, family = "binomial")  
 print(summary(mylogit))

Call:

```
glm(formula = LowBirthWeight ~ ., family = "binomial", data = train)
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-3.971e-06	3.971e-06	3.971e-06	3.971e-06	3.971e-06

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	2.557e+01	2.283e+06	0	1
headcircumference	2.394e-07	1.553e+05	0	1
length	-4.789e-07	7.673e+04	0	1
Birthweight	-7.068e-07	9.882e+04	0	1
Gestation	2.199e-07	3.224e+04	0	1
smoker	-4.695e-07	1.471e+05	0	1
motherage	-6.980e-08	2.662e+04	0	1
mnocig	1.314e-08	5.185e+03	0	1
mheight	4.298e-08	4.016e+04	0	1
mppwt	1.170e-09	6.293e+03	0	1
fage	2.257e-08	2.161e+04	0	1
fedyrs	-1.256e-08	2.780e+04	0	1
fnocig	-1.218e-08	3.734e+03	0	1
fheight	4.529e-09	2.751e+04	0	1
lowbwt	-5.113e+01	2.686e+05	0	1
mage35	1.135e-07	3.308e+05	0	1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2.6662e+01 on 28 degrees of freedom  
 Residual deviance: 4.5733e-10 on 13 degrees of freedom  
 AIC: 32

Number of Fisher Scoring iterations: 24

```
In [13]: pred = predict(mylogit,
                      newdata = test,
                      type = "response")

pred_value <- ifelse(pred > 0.5, "Normal", "Low")
print("Testdata admit vs predict:")
result <- data.frame(Actual = test$LowBirthWeight, pred_value)
print(result)

[1] "Testdata admit vs predict:"
  Actual pred_value
 6  Normal     Normal
 8  Normal     Normal
10 Normal     Normal
15 Normal     Normal
17 Normal     Normal
19    Low       Low
21 Normal     Normal
25 Normal     Normal
27 Normal     Normal
28 Normal     Normal
34 Normal     Normal
37 Normal     Normal
40 Normal     Normal
```

```
In [14]: # SOLUTION 1
misClasificError <- mean(pred_value != test$LowBirthWeight)
print(paste('Accuracy s2: ', 1-misClasificError))

[1] "Accuracy s2: 1"
```

```
In [15]: names(test)

'headcircumference' 'length' 'Birthweight' 'Gestation' 'smoker' 'motherage' 'mnocig'
'mheight' 'mppwt' 'fage' 'fedyrs' 'fnocig' 'fheight' 'lowbwt' 'mage35' 'LowBirthWeight'
```

```
In [16]: # predict new
# sample: (12, 18, 4.5, 35, 1, 41, 7, 65, 125, 37, 14, 25, 68, 1, 1)
y1 <- predict(mylogit,
              newdata = data.frame(headcircumference = c(12),
                                    length = c(18),
                                    Birthweight = c(4.5),
                                    Gestation = c(35),
                                    smoker = c(1),
                                    motherage = c(41),
                                    mnocig = c(7),
                                    mheight = c(65),
                                    mppwt = c(125),
                                    fage = c(37),
                                    fedyrs = c(14),
                                    fnocig = c(25),
                                    fheight = c(68),
                                    lowbwt = c(1),
                                    mage35 = c(1)
              ),
              type='response')
y1 <- ifelse(y1 > 0.5, 1, 0)
print("results:")
print(y1)
```

```
[1] "results:"
1
0
```