CLW_R_Learning：Logistic Regression

Jason Wang

Tuesday, April 21, 2015

1. Little Introduction to Generalized Linear Models(GLM)

There are three components in the GLM:

- Random Component — the response variable Y and an associated probability distribution.
- Systematic Component — the explanatory variables $x_1, ..., x_p$
- Link Function — the functional relationship between $x_1, ..., x_p$ and $E(Y)$

Y	Link function
$N(\mu, \sigma^2)$	$E(Y)$
$Poisson(\lambda)$	$log(E(Y))$
$Binomial(n, p)$	$log\frac{E(Y)}{n - E(Y)}$

2. Logistic Regression

First, we introduce the logstic function by a illustration.

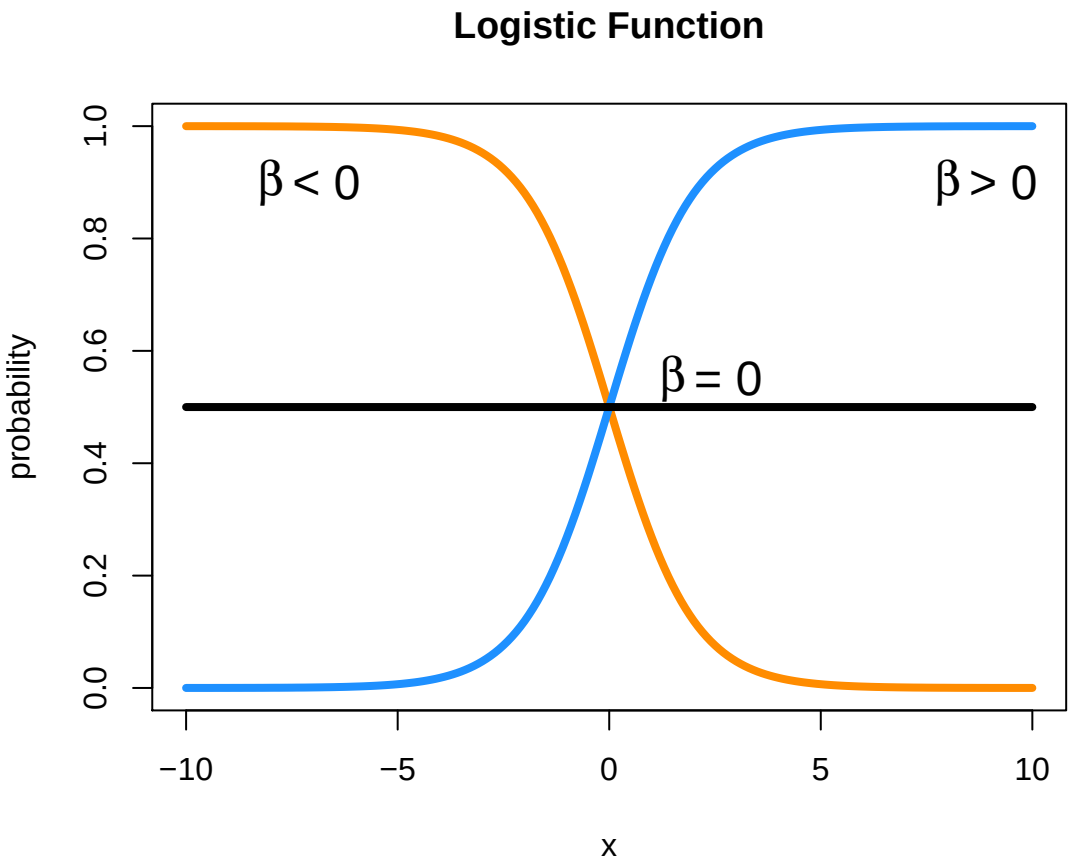
$$p(x) = \frac{e^{\alpha+\beta x}}{1 + e^{\alpha+\beta x}}$$

```
logistic <- function(x, alpha, beta){
    exp(alpha + beta*x)/(1 + exp(alpha + beta*x))
}
```

```
x <- seq(-10, 10, by=0.1)
plot(x, logistic(x, 0, -1), type="l",
     main="Logistic Function", col="darkorange",
     xlab="x", ylab="probability", lwd=4)
lines(x, logistic(x, 0, 1), type="l",
      col="dodgerblue", lwd=4)
```

```
text(8, 0.9, expression(beta), cex=1.5); text(-8, 0.9, expression(beta), cex=1.5)
text(9, 0.9, "  > 0", cex=1.5); text(-7, 0.9, "  < 0", cex=1.5)

lines(x, logistic(x, 0, 0), type="l", lwd=4)
text(1.5, 0.55, expression(beta), cex=1.5)
text(2.5, 0.55, "  = 0", cex=1.5)
```



If β is bigger than 0, then the probability will increase as x goes up; on the other hand, if β is less than 0, the probability will decrease as x goes up. Finally, when β is equal to 1, the probability will be 0.5 given α is 0.

With the above illustration, it gives us a hint or implication that whether we can utilize this kind of function to fit a model on the binary response data. Hence, here come to the logistic regression.

$$E(y|X) = 1 \times P(y = 1|X) + 0 \times P(y = 0|X) = P(y = 1|X)$$

We model $E(y|X)$ using the logistic function that gives outputs between 0 and 1 for all values of X.

$$p = E(y|X) = \frac{e^{\alpha+\beta_1 x_1+\beta_2 x_2+...+\beta_p x_p}}{1+e^{\alpha+\beta_1 x_1+\beta_2 x_2+...+\beta_p x_p}}$$

With some manipulation, we can get

$$\frac{p}{1-p} = e^{\alpha+\beta_1 x_1+\beta_2 x_2+...+\beta_p x_p}$$

$\frac{p}{1-p}$ is called the odds and take on any value from 0 to ∞. If the odds is high, the probability is high, and vice versa.

Take log for both side

$$log(\frac{p}{1-p}) = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$$

From the left-hand side, we get a function called *logit* function. $logit(p) = log(\frac{p}{1-p})$. The logit function is exactly the **link function** between $E(Y)$ and $x_1, ..., x_p$.

- Interpretation

The interpretation of the coefficients x_i will be totally different from the linear regression. In logistic regression, the interpretation is that given other variables fix constant, increasing x_i by one unit change the log-odds of $E(Y)$ or p by β_i unit or, equivalently, multiplies the odds of p by e^{β_i}. Most importantly, The amount that p changes due to a one-unit change in x_i will depend on the current value of x_i.

- Estimation of Coefficients

In logistic regression, we use *maximum likelihood estimation* to estimate coefficients. Recall that our Y is from the binomial distribution (can be view as bernoulli trials when solving the likelihood function), assuming n data, we can use the *likelihood function*:

$$l(\alpha, \beta_1, \beta_2, ..., \beta_p) = \prod_{i=1}^n p(y_i|x_i)$$

$$= \prod_{i=1}^n [\frac{e^{\alpha+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}}}{1+e^{\alpha+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}}}]^{y_i} [\frac{1}{1+e^{\alpha+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}}}]^{1-y_i}$$

and our goal is to find out the coefficients $\alpha, \beta_1, \beta_2, ..., \beta_p$ which maximize the likelihood function. Normally, we will transformation (log) the likelihood to *deviance*:

$$D = -2[\sum_{i=1}^n y_i log(p_i) + \sum_{i=1}^n (1-y_i) log(1-pi)]$$

$$\text{where} \quad p_i = \frac{e^{\alpha+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}}}{1+e^{\alpha+\beta_1 x_{1i}+\beta_2 x_{2i}+...+\beta_p x_{pi}}}$$

After the transformation, the goal change to find the parameters that minimize the deviance. Fortunately, we can use R to solve the coefficients effectively. (Note: In R, its output is -2log(l), i.e. Deviance, while in SAS, the output is log(l), i.e. Log likelihood function)

```r
#Deviance illustration
null_deviance_bin <- function(y){
    p <- mean(y)
    D <- -2*(sum(y*log(p) + (1 - y)*log(1 - p)))
    D
}
```

```r
y <- rbinom(100, 1, 0.8)
x <- rnorm(100)
model <- glm(y ~ x, family=binomial)
#Without variable
summary(model)$null.deviance
```

```
## [1] 100.0805
```

```r
null_deviance_bin(y)
```

```
## [1] 100.0805
```

```r
#With variable
summary(model)$deviance
```

```
## [1] 99.9715
```

```r
beta <- coef(model)
model_deviance <- function(y, x, beta){
    e <- exp(cbind(rep(1, length(x)), x)%*%beta)
    p <- e/(1 + e)
    D <- -2*(sum(y*log(p) + (1 - y)*log(1 - p)))
    D
}
model_deviance(y, x, beta)
```

```
## [1] 99.9715
```

The idea of deivance is really similar to sum of square in regression. Summary:

Regression model → analysis of variance (compare ”sum of square”)
Generalized linear model → analysis of deviance (compare ”deviance”)

- Three Main Types of Statistical Tests for GLMs

1. Wald test

$$\text{the test statsitic} \quad Z = \frac{\hat{\beta}}{ASE} \quad \xrightarrow{D} \quad N(0,1)$$

The summary function to glm will automatically provide us the Wald test.

```
coef(summary(model))
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) 1.39165339  0.2511429 5.5412814 3.002662e-08
## x           0.08161013  0.2475777 0.3296344 7.416762e-01
```

2. Likelihood ratio test

$$\text{the likelihood-ratio statistic} = -2log(\frac{l_o}{l_1}) = -2(logl_0 - logl_1) = -2(L_0 - L_1) \quad \xrightarrow{D} \quad \chi^2_1$$

```
library(lmtest)
#Likelihood ratio test
lrtest(model)
```

```
## Likelihood ratio test
##
## Model 1: y ~ x
## Model 2: y ~ 1
##   #Df  LogLik Df Chisq Pr(>Chisq)
## 1   2 -49.986
## 2   1 -50.040 -1 0.109     0.7413
```

The log likelihood function here is equal to the deviance divided by 2. Also we can use the anova function to do likelihood ratio test.

```
anova(model, test="LRT")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
```

```
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                    99    100.080
## x     1  0.10898        98     99.971   0.7413
```

3. Score test

the efficient score statistics $\xrightarrow{D}$ χ^2_1

We can use the anova function to do likelihood ratio test.

```
anova(model, test="Rao")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev     Rao Pr(>Chi)
## NULL                    99    100.080
## x     1  0.10898        98     99.971 0.10878   0.7415
```

In generally, the three test would give you the same conclusion for significance. If it is not, the suggestion is to collect more data. However, if you cannot get more data, then take the likelihood ratio test because LR test is more conservative.

3. Multinomial Logistic Regression

Also, we can generalize the method to multinomial situation. Assume that we have g possible outcomes with probabilities $P[y = k] = p_k, k = 1, 2, ..., g$.

$$p_1 = p[y = 1] = \frac{1}{1 + \sum_{i=2}^{g} e^{\alpha_i + x\beta_i}}$$
$$p_k = p[y = k] = \frac{e^{\alpha_k + x\beta_k}}{1 + \sum_{i=2}^{g} e^{\alpha_i + x\beta_i}} \text{ for } k = 2, ..., g$$

The sum of $p_1, p_2, ..., p_g$ is equal to 1. The category 1 here is our standard category while any other group could be use instead.

The log odds interpretation of the logistic regression model still applies, as

$$log(\frac{p_k}{p_1}) = \alpha_k + x\beta_k \text{ for } k = 2, ...g.$$

However, the interpretation become more complicated. Changing the the explanatory variable x_i by on unit changes the odds of getting an outcome from group k relative to getting an outcome from gropu 1. (Note the log odds now become p_k v.s. p_1) For instance, if β_k = 1.5, per unit change in variable x_i increase the odds by mutiplying by $e^{1.5}$ = 4.48. Similarly, per unit change in x changes the odds of getting an outcome from group k relative to getting an outcome from group r by factor $e^{\beta_k - \beta_r}$ because

$$\frac{p_k}{p_r} = \frac{e^{\alpha_k + x\beta_k}}{e^{\alpha_r + x\beta_r}} = e^{\alpha k - \alpha r + x(\beta_k - \beta r)}$$
$$log(\frac{p_k}{p_1}) = \alpha_k - \alpha_r + x(\beta_k - \beta r)$$

The interpreation is so complex, which may be the reason why the textbook skips this session. We generate a fictional data to run the multinomail logistic regression.

```
#5 categories
y <- gl(5, 20)
#normal which increases the mean
x <- rnorm(100, rep(1:5, each=20))

#Package that can run multinomial logistic regression
library(VGAM)


m_model <- vglm(y ~ x, multinomial)
summary(m_model)
```

```
##
## Call:
## vglm(formula = y ~ x, family = multinomial)
##
## Pearson residuals:
##                         Min      1Q   Median         3Q   Max
## log(mu[,1]/mu[,5]) -2.042 -0.2538 -0.01265 -0.0001647 4.145
## log(mu[,2]/mu[,5]) -2.490 -0.4332 -0.05403 -0.0016787 8.981
## log(mu[,3]/mu[,5]) -2.798 -0.3486 -0.16498 -0.0396555 3.870
## log(mu[,4]/mu[,5]) -2.320 -0.4776 -0.08289 -0.0038809 1.875
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  13.8546     2.5739   5.383 7.34e-08 ***
## (Intercept):2  12.8573     2.5357   5.070 3.97e-07 ***
## (Intercept):3   7.9367     2.1275   3.730 0.000191 ***
## (Intercept):4   3.6319     1.7656   2.057 0.039682 *
## x:1            -4.7399     0.8378  -5.658 1.54e-08 ***
## x:2            -3.9798     0.7696  -5.171 2.33e-07 ***
## x:3            -1.9693     0.5199  -3.788 0.000152 ***
## x:4            -0.8106     0.3877  -2.091 0.036559 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of linear predictors:  4
##
## Names of linear predictors:
## log(mu[,1]/mu[,5]), log(mu[,2]/mu[,5]), log(mu[,3]/mu[,5]), log(mu[,4]/mu[,5])
##
## Dispersion Parameter for multinomial family:   1
##
## Residual deviance: 200.8219 on 392 degrees of freedom
##
## Log-likelihood: -100.411 on 392 degrees of freedom
##
## Number of iterations: 7
```

```
round(fitted(m_model), 2)
```

```
##        1    2    3    4    5
## 1   0.05 0.18 0.50 0.21 0.06
## 2   0.71 0.28 0.00 0.00 0.00
## 3   0.55 0.43 0.02 0.00 0.00
## 4   0.06 0.19 0.50 0.20 0.06
## 5   0.76 0.24 0.00 0.00 0.00
## 6   0.69 0.31 0.00 0.00 0.00
## 7   0.49 0.46 0.04 0.00 0.00
## 8   0.66 0.34 0.01 0.00 0.00
## 9   0.74 0.26 0.00 0.00 0.00
## 10  0.67 0.32 0.00 0.00 0.00
## 11  0.52 0.45 0.03 0.00 0.00
## 12  0.49 0.46 0.04 0.00 0.00
## 13  0.35 0.50 0.13 0.01 0.00
## 14  0.37 0.50 0.12 0.01 0.00
## 15  0.52 0.45 0.03 0.00 0.00
```

```
## 16  0.75 0.25 0.00 0.00 0.00
## 17  0.54 0.44 0.03 0.00 0.00
## 18  0.66 0.34 0.01 0.00 0.00
## 19  0.26 0.47 0.23 0.04 0.01
## 20  0.51 0.45 0.03 0.00 0.00
## 21  0.42 0.49 0.08 0.01 0.00
## 22  0.33 0.50 0.15 0.02 0.00
## 23  0.52 0.45 0.03 0.00 0.00
## 24  0.37 0.50 0.12 0.01 0.00
## 25  0.73 0.27 0.00 0.00 0.00
## 26  0.18 0.40 0.33 0.07 0.01
## 27  0.14 0.35 0.39 0.09 0.02
## 28  0.11 0.30 0.44 0.13 0.03
## 29  0.44 0.49 0.07 0.00 0.00
## 30  0.71 0.29 0.00 0.00 0.00
## 31  0.30 0.49 0.19 0.02 0.00
## 32  0.65 0.34 0.01 0.00 0.00
## 33  0.36 0.50 0.12 0.01 0.00
## 34  0.49 0.47 0.04 0.00 0.00
## 35  0.34 0.50 0.14 0.02 0.00
## 36  0.00 0.01 0.29 0.41 0.29
## 37  0.55 0.43 0.02 0.00 0.00
## 38  0.58 0.41 0.02 0.00 0.00
## 39  0.30 0.49 0.18 0.02 0.00
## 40  0.32 0.50 0.16 0.02 0.00
## 41  0.02 0.08 0.47 0.31 0.12
## 42  0.26 0.47 0.23 0.03 0.00
## 43  0.00 0.00 0.19 0.42 0.39
## 44  0.00 0.00 0.09 0.37 0.54
## 45  0.00 0.00 0.19 0.42 0.39
## 46  0.02 0.10 0.49 0.28 0.10
## 47  0.20 0.42 0.32 0.06 0.01
## 48  0.07 0.22 0.49 0.18 0.05
## 49  0.07 0.21 0.49 0.18 0.05
## 50  0.16 0.38 0.37 0.08 0.02
## 51  0.10 0.28 0.45 0.14 0.03
## 52  0.39 0.50 0.10 0.01 0.00
## 53  0.05 0.18 0.50 0.21 0.06
## 54  0.00 0.00 0.06 0.34 0.60
## 55  0.00 0.02 0.34 0.40 0.24
## 56  0.00 0.01 0.29 0.41 0.29
## 57  0.01 0.04 0.41 0.37 0.18
## 58  0.03 0.11 0.50 0.27 0.09
## 59  0.18 0.40 0.33 0.07 0.01
## 60  0.04 0.14 0.50 0.24 0.08
```

```
## 61  0.00 0.01 0.25 0.42 0.32
## 62  0.00 0.00 0.10 0.38 0.52
## 63  0.06 0.20 0.50 0.19 0.05
## 64  0.00 0.00 0.14 0.40 0.46
## 65  0.00 0.02 0.33 0.40 0.25
## 66  0.00 0.01 0.26 0.42 0.31
## 67  0.00 0.00 0.02 0.26 0.72
## 68  0.00 0.00 0.07 0.34 0.59
## 69  0.00 0.01 0.29 0.41 0.28
## 70  0.02 0.09 0.48 0.30 0.11
## 71  0.00 0.01 0.30 0.41 0.27
## 72  0.00 0.01 0.28 0.41 0.29
## 73  0.01 0.06 0.45 0.33 0.14
## 74  0.00 0.01 0.26 0.42 0.31
## 75  0.00 0.00 0.10 0.38 0.51
## 76  0.02 0.09 0.49 0.29 0.11
## 77  0.00 0.01 0.29 0.41 0.28
## 78  0.00 0.00 0.16 0.41 0.42
## 79  0.00 0.00 0.17 0.41 0.41
## 80  0.01 0.04 0.40 0.37 0.19
## 81  0.00 0.02 0.34 0.40 0.24
## 82  0.00 0.00 0.02 0.25 0.72
## 83  0.00 0.00 0.01 0.20 0.78
## 84  0.00 0.00 0.06 0.34 0.60
## 85  0.00 0.02 0.36 0.39 0.22
## 86  0.00 0.00 0.02 0.22 0.76
## 87  0.00 0.00 0.20 0.42 0.37
## 88  0.00 0.00 0.21 0.42 0.37
## 89  0.00 0.00 0.01 0.15 0.84
## 90  0.08 0.24 0.48 0.16 0.04
## 91  0.00 0.01 0.27 0.42 0.31
## 92  0.00 0.03 0.38 0.38 0.20
## 93  0.00 0.01 0.27 0.42 0.30
## 94  0.00 0.00 0.06 0.34 0.60
## 95  0.00 0.00 0.05 0.32 0.63
## 96  0.00 0.00 0.20 0.42 0.38
## 97  0.00 0.00 0.02 0.22 0.77
## 98  0.00 0.00 0.11 0.39 0.50
## 99  0.00 0.01 0.24 0.42 0.33
## 100 0.00 0.00 0.02 0.24 0.74
```

```
(fit <- unlist(apply(round(fitted(m_model), 2), 1,
            function(x) as.numeric(which(x == max(x))[1]))))
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   3   1   1   3   1   1   1   1   1   1   1   1   2   2   1   1   1   1
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   2   1   2   2   1   2   1   2   3   3   2   1   2   1   2   1   2   4
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   1   2   2   3   2   4   5   4   3   2   3   3   2   3   2   3   5
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   4   4   3   3   2   3   4   5   3   5   4   4   5   5   4   3   4   4
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   3   4   5   3   4   5   4   3   4   5   5   5   4   5   4   4   5   3
##  91  92  93  94  95  96  97  98  99 100
##   4   3   4   5   5   4   5   5   4   5
```

```
table(true=y, predict=fit)
```

```
##     predict
## true  1  2  3  4  5
##    1 15  3  2  0  0
##    2  7 10  2  1  0
##    3  0  5  9  4  2
##    4  0  0  5  9  6
##    5  0  0  2  8 10
```

4. Example on the Textbook

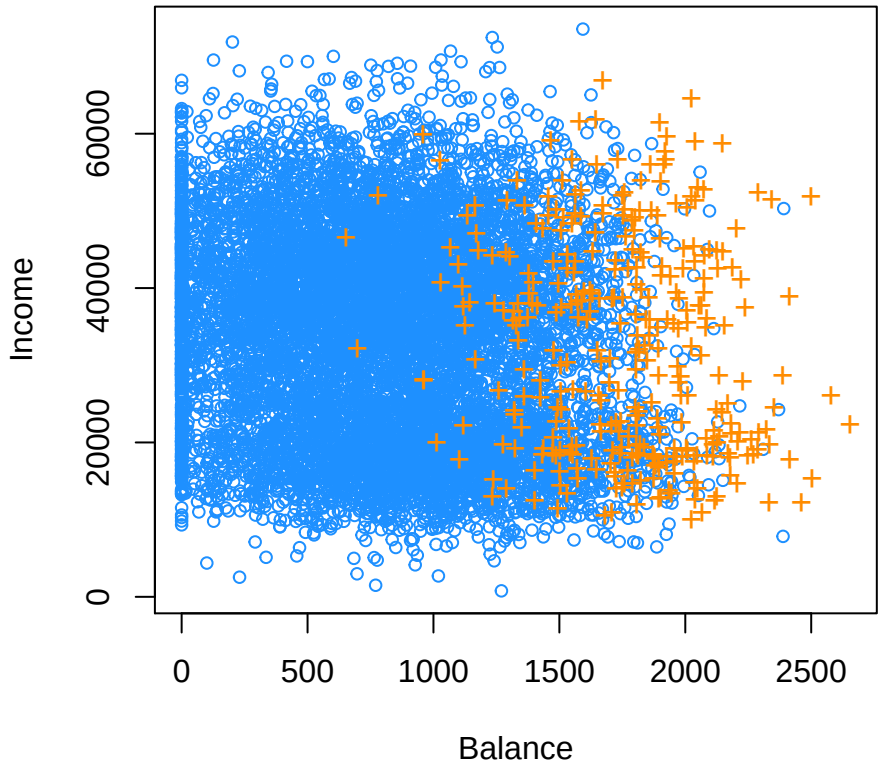
```
library(ISLR)
summary(Default)
```

```
##  default    student       balance           income
##  No :9667   No :7056   Min.   :   0.0   Min.   :  772
##  Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
##                        Median : 823.6   Median :34553
##                        Mean   : 835.4   Mean   :33517
##                        3rd Qu.:1166.3   3rd Qu.:43808
##                        Max.   :2654.3   Max.   :73554
```

```r
with(Default, plot(balance, income, type="n", xlab="Balance", ylab="Income"))
split_data <- split(Default, Default$default)

no_df <- split_data[[1]]
df <- split_data[[2]]

with(no_df, points(balance, income, pch=1, cex=0.8, col="dodgerblue"))
with(df, points(balance, income, pch="+", cex=1.2, col="darkorange"))
```




```r
library(car)
```

```
##
## Attaching package: 'car'
##
## The following object is masked from 'package:VGAM':
##
##     logit
```

```
Default$default <- as.numeric(as.character(
    recode(Default$default, "'Yes'='1'; 'No'='0'")))
```

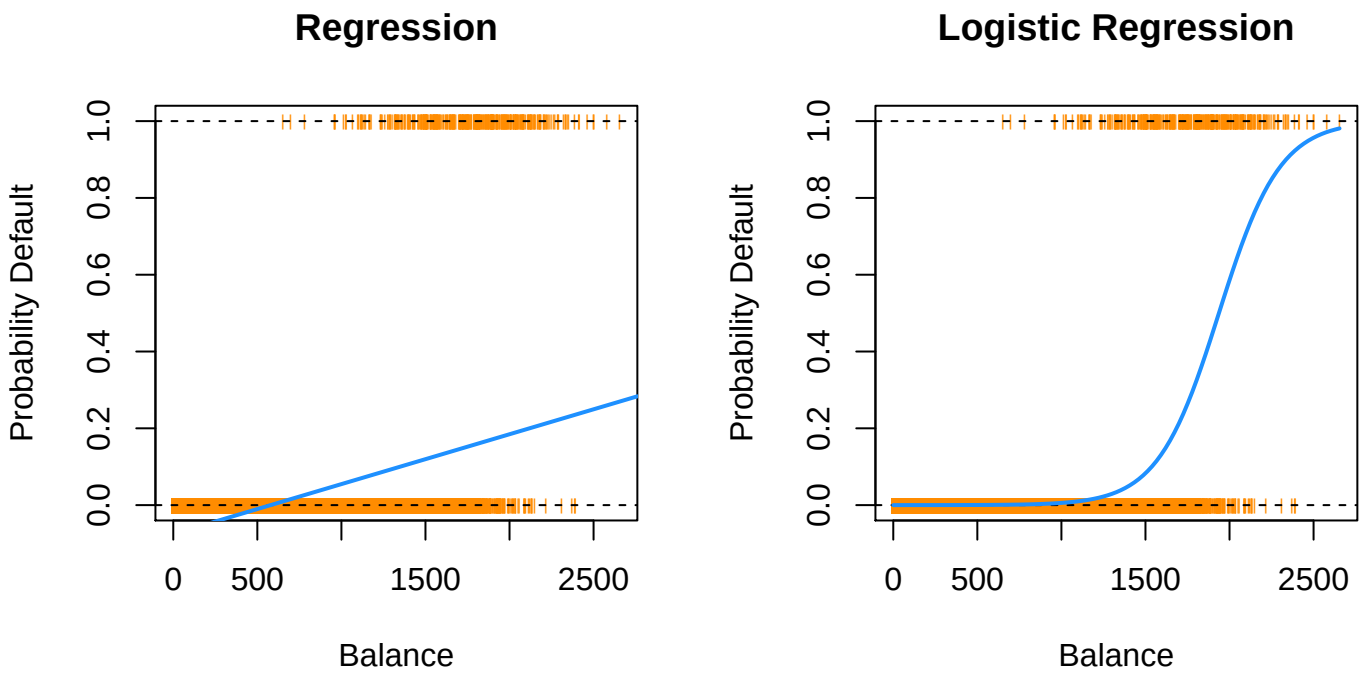
```
#Page 131
#Regression
model1 <- lm(default ~ balance, data=Default)

#Logistic Regression
model2 <- glm(default ~ balance, data=Default, family=binomial)
```

```
par(mfrow=c(1, 2))
with(Default, plot(balance, default, col="darkorange",
                   pch="|", cex=0.5, main="Regression",
                   xlab="Balance", ylab="Probability Default"))
abline(h=c(1, 0), lty=2)
abline(model1, col="dodgerblue", lwd=2)


with(Default, plot(balance, default, col="darkorange",
                   pch="|", cex=0.5, main="Logistic Regression",
                   xlab="Balance", ylab="Probability Default"))
abline(h=c(1, 0), lty=2)

curve(predict(model2, data.frame(balance=x), type="resp"),
      add=TRUE, lwd=2, col="dodgerblue")
```

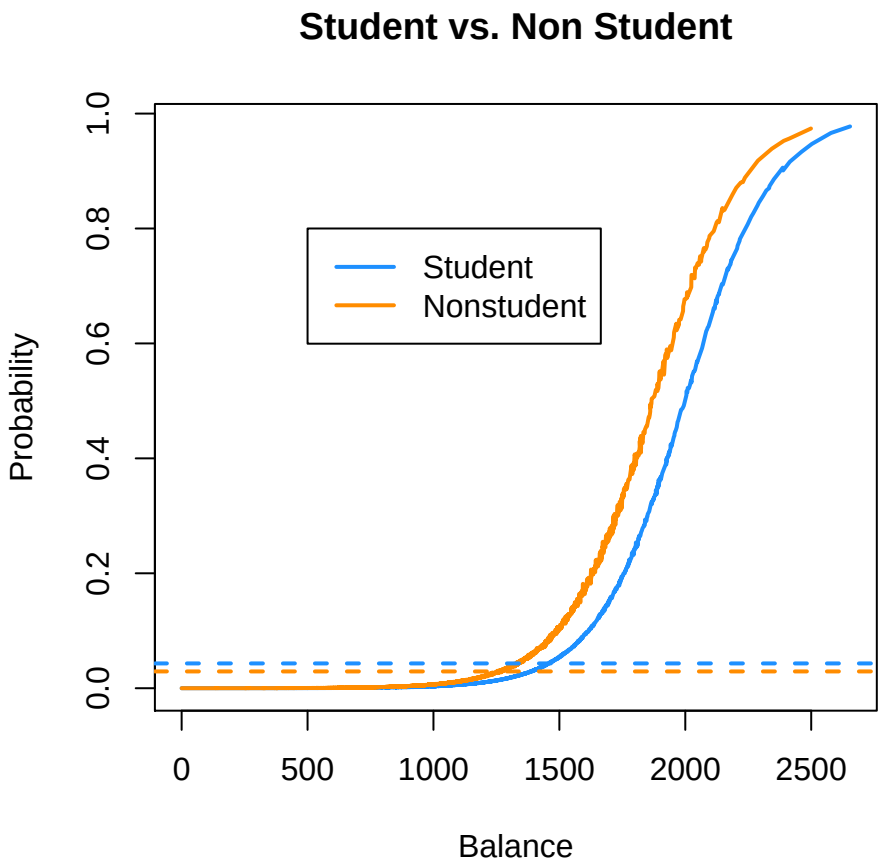



```
#balance
model2 <- glm(default ~ balance, data=Default, family=binomial)
#student
model3 <- glm(default ~ student, data=Default, family=binomial)
#all
model4 <- glm(default ~ balance + income + student, data=Default, family=binomial)
```

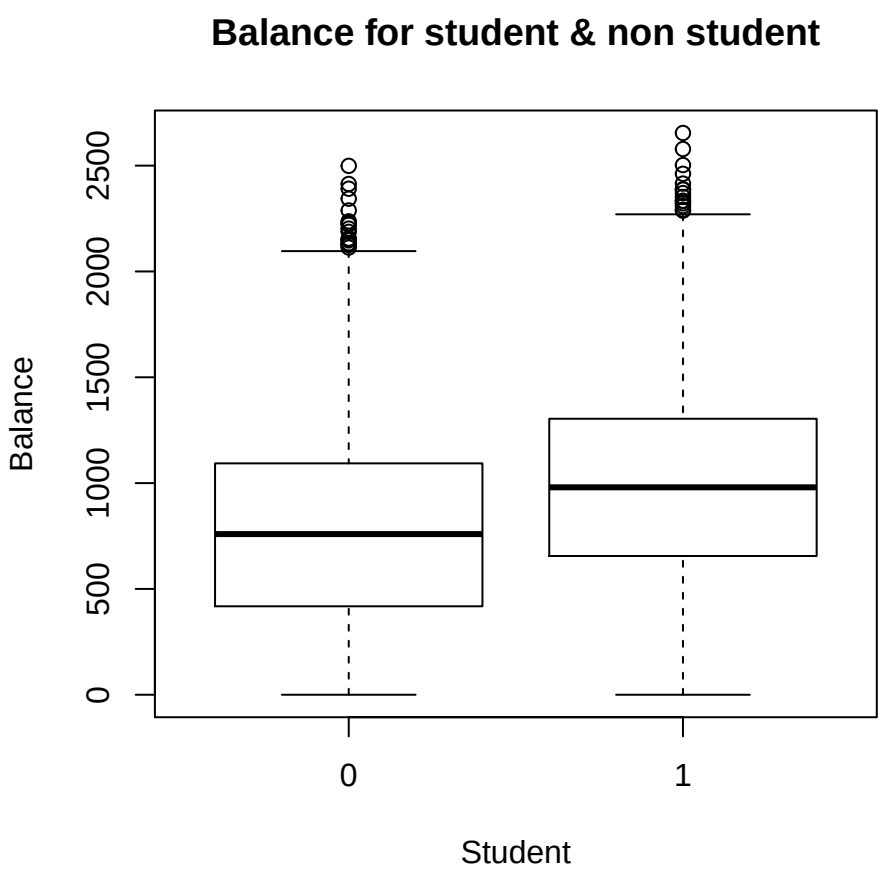
```
Default$student <- as.numeric(as.character(recode(Default$student, "'Yes'='1'; 'No'='0'

fit_p <- fitted(model4)
D <- cbind(Default, fit_p)
D <- D[order(D$balance), ]
average <- tapply(Default$default, Default$student, mean)
plot(D$balance[D$student == 1], D$fit_p[D$student == 1],
     main="Student vs. Non Student",
     xlab="Balance", ylab="Probability",
     type="l", col="dodgerblue", lwd=2)
lines(D$balance[D$student == 0], D$fit_p[D$student == 0],
      type="l", col="darkorange", lwd=2)
abline(h=average[1], col="darkorange", lty=2, lwd=2)
abline(h=average[2], col="dodgerblue", lty=2, lwd=2)
legend(500, 0.8, lty=1, lwd=2, col=c("dodgerblue","darkorange"),
```

```
       legend=c("Student", "Nonstudent"))
```




```
boxplot(balance ~ student, data=Default,
        main="Balance for student & non student",
        xlab="Student", ylab="Balance")
```


Balance for student & non student
Balance
0
500
1000
1500
2000
2500
0
1
Student