Logistic Regression

Chih-Hui Wang (Jason)

Tuesday, April 21, 2015; Revised: March 24, 2016

1. 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 $E(Y)$

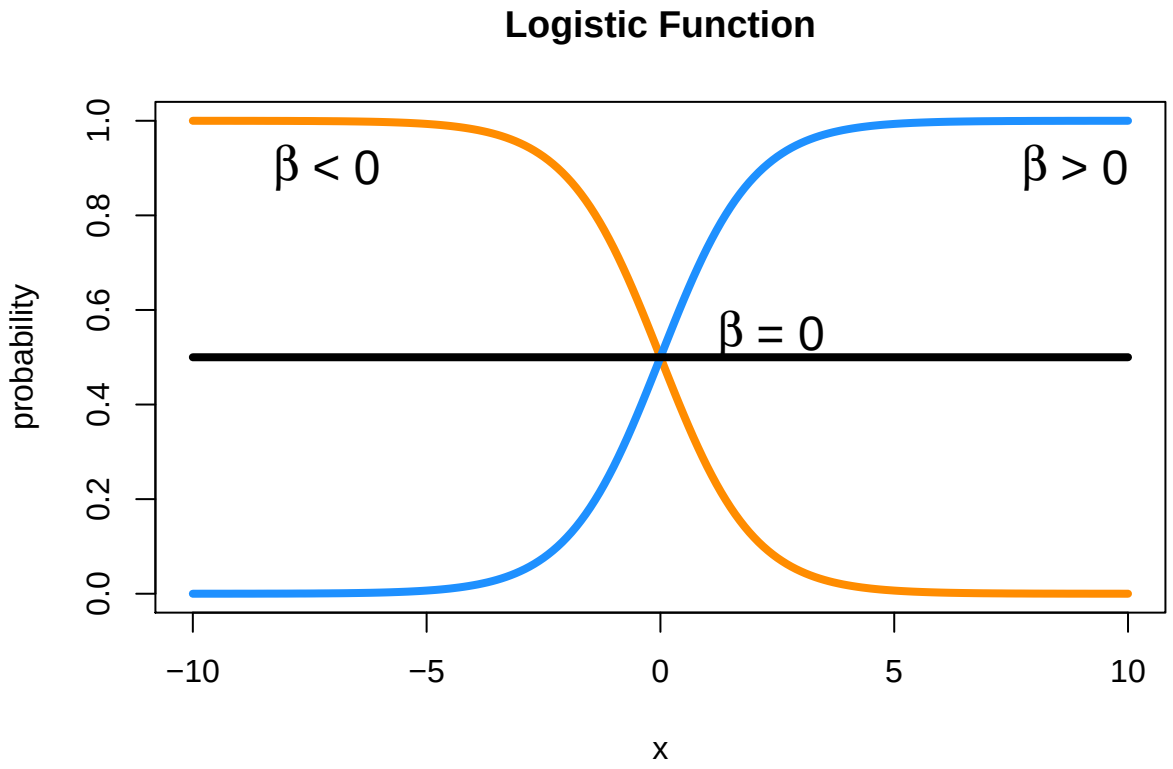
Y	Link function
$N(\mu, \sigma^2)$	$E(Y)$
$Poisson(\lambda)$	$log(E(Y))$
$Binomial(n, p)$	$log\frac{E(Y)}{n-E(Y)}$

2. Logistic Regression

First, we introduce the logistic function or sigmoid function.

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

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

# plot
x <- seq(-10, 10, by=0.1)

# Beta < 0
plot(x, logistic(x, 0, -1), type="l", main="Logistic Function", col="darkorange",
     xlab="x", ylab="probability", lwd=4)

# Beta > 0
lines(x, logistic(x, 0, 1), type="l", col="dodgerblue", lwd=4)

# Beta = 0
lines(x, logistic(x, 0, 0), type="l", lwd=4)

# Add text
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)
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 0, the probability will always be 0.5 given α is 0.

Since the function can map any real value to the interval $[0, 1]$, it gives us a thought that perhaps we can utilize this property to fit a model to the binary response data, which is the idea of 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:

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

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 of Logistic Regression

The interpretation of the coefficients x_i is entirely different from the linear regression. In the logistic regression, the interpretation is that given other variables 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, assuming n data, we can use the *likelihood function*:

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

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

and our goal is to find out the coefficients $\alpha, \beta_1, \beta_2, ..., \beta_p$ which maximize the likelihood function. Usually, we will use log likelihood function:

$$l = \sum_{i=1}^{n} y_i log(p_i) + \sum_{i=1}^{n} (1-y_i) log(1-pi)$$

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

1. Gradient Ascent

So now we have a optimization problem: we need to find the α, β that maximize the log likelihood. We can use gradient ascent. We rewrite the log likelihood function into matrix form

$$l = \sum_{i=1}^{n} y_i \log s(X_i\beta) + (1-y_i) \log (1 - s(X_i\beta))$$

where β is a vector with length $n+1$ (including α) and X_i is the obervation i. We need to find the gradient of the log-likelihood function, let $s_i = s(X_i\beta)$,

$$\Delta l = \sum_{i=1}^{n} y_i \frac{\Delta s_i}{s_i} - (1-y_i) \frac{\Delta s_i}{1-s_i} \tag{3}$$

$$= \sum_{i=1}^{n} (\frac{y_i}{s_i} - \frac{1-y_i}{1-s_i}) s_i(1-s_i) X_i \tag{4}$$

$$= \sum_{i=1}^{n} (y_i - s_i) X_i = X^T(y-s) \tag{5}$$

Note that $s'(x) = s(x)(1-s(x))$. Therefore, the gradient ascent rule is

$$\beta_{t+1} = \beta_t + \epsilon \sum_{i=1}^{n} (y_i - s(X_i\beta_t)) X_i$$

Keep updating the parameters until the difference is smaller than a certain threshold.

2. Newton Method

Idea: You are at point v. Approximate the function near v by a quadratic function and jump to its unique critical point. Repeat until convergence.
We use Taylor series: $\Delta l(\beta) = \Delta l(v) + (\Delta^2 l(v))(\beta - v) + O(|\beta - v|^2)$ and find the critical point by setting $\Delta l(\beta) = 0$. We get $w = v - (\Delta^2 l(v))^{-1}\Delta l(v))$. Here is the summary of the algorithm:

```
# Algorithm: picking starting point w
# repeat until convergence:
#       e <- solution to the linear system (\Delta^2 l(\beta)e=-\Delta l(\beta))
#       beta <- beta + e
```

In logistic regression, we need to calculate the hessian of the log likelihood function.

$$\Delta^2 l = -\sum_{i=1}^{n} s_i(1 - s_i)X_i X_i^T = -X^T \Omega X$$

we solve e in the normal equation $(X^T \Omega X)e = X^T(y - s)$ where Ω is a diagonal matrix with components $s_i(1 - s_i)$.

When you fit the GLM in the statistical software, they will output the deviance for you which is defined as $D = -2l$. (Note: In R, its output is -2log(l), i.e. Deviance, while in SAS, the output is log(l), i.e. Log likelihood function) The idea of deivance is really similar to sum of square in regression:

Regression model $\rightarrow$ analysis of variance (compare "sum of square")

Generalized linear model $\rightarrow$ analysis of deviance (compare "deviance")

Hypothesis Testing: three Main Types of Statistical Tests for GLMs

1. Wald test
   The Wald test statsitic
   $$Z = \frac{\hat{\beta}}{ASE} \xrightarrow{D} N(0, 1)$$
   The summary function to glm will automatically provide us the Wald test.

```
# Simulated Data
y <- rbinom(100, 1, 0.8)
x <- rnorm(100)
model <- glm(y ~ x, family=binomial)

coef(summary(model))
```

```
             Estimate Std. Error  z value     Pr(>|z|)
(Intercept) 1.7336698   0.280701 6.176215 6.565637e-10
x           0.1433059   0.280916 0.510138 6.099548e-01
```

2. Likelihood ratio test
   The likelihood-ratio statistic
   $$l = -2log(\frac{l_o}{l_1}) = -2(logl_0 - logl_1) = -2(L_0 - L_1) \xrightarrow{D} \chi_1^2$$

```r
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 -42.139
2   1 -42.271 -1 0.2638     0.6075
```

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.

```r
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     84.542
x     1  0.26385        98     84.278   0.6075
```

3. Score (Rao) test The efficient score statistics

$$S \xrightarrow{D} \chi^2_1$$

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

```r
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     84.542
x     1  0.26385        98     84.278 0.26105   0.6094
```

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. Linear Discriminant Analysis vs. Logistic Regression

Advantage of LDA:
a. For well separated data, LDA is stable while logistic regression is relatively unstable.
b. For more than two classes, LDA is easy and elegant to generalize; logistic regression needs modifying.
c. Slighly more accurate when classes are nearly normal, especially if n is small.

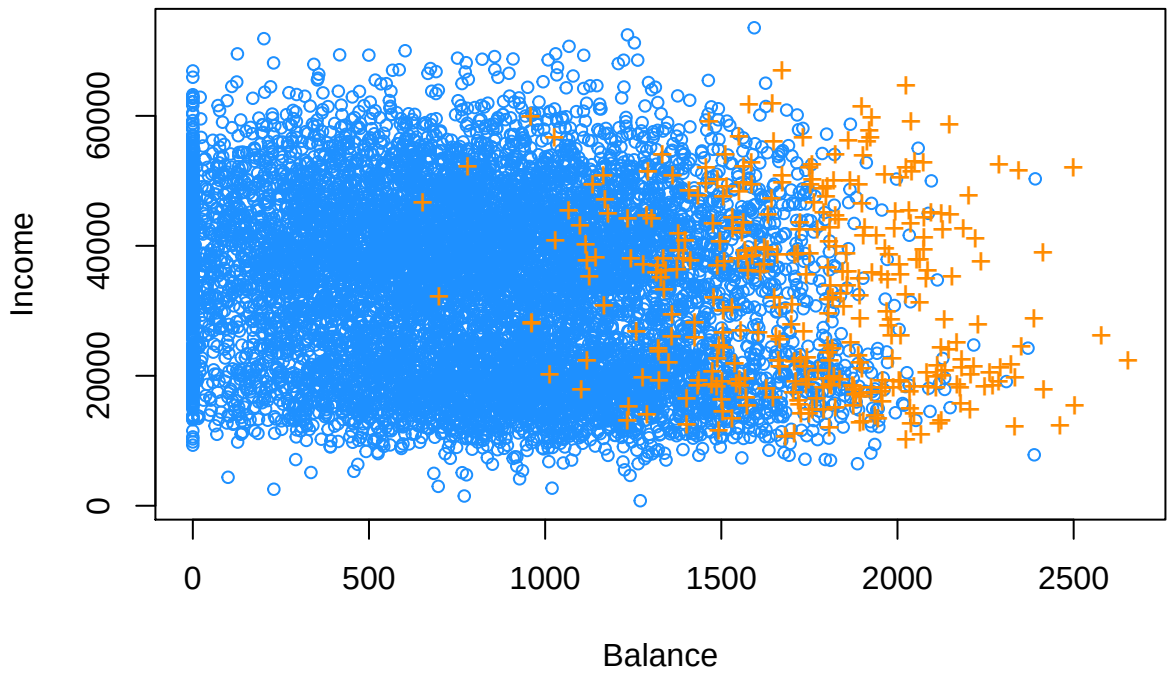
Advantage of logistic regression:
a. More emphasis on the decision boundary.
b. Less sensitive to outliers.
c. More robust on some non-Gaussian distribution.

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
```

```
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"))
```

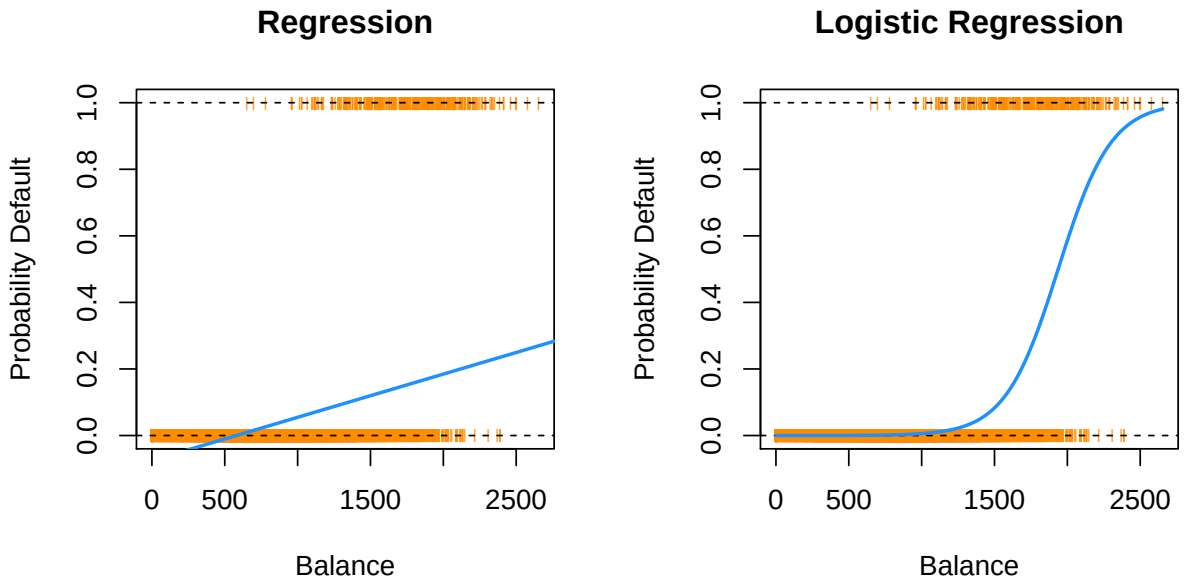



```
library(car)
Default$default <- as.numeric(as.character(
    recode(Default$default, "'Yes'='1'; 'No'='0'")))
```

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

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

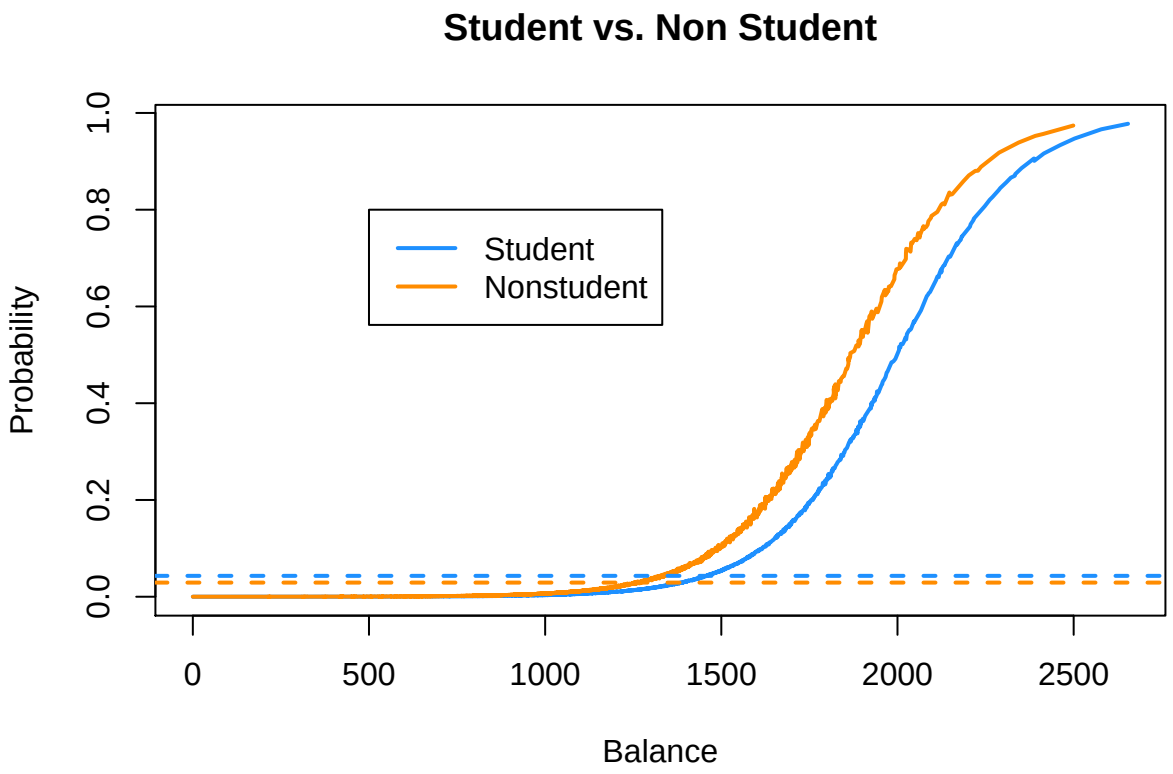
```
par(mfrow=c(1, 2))
with(Default, plot(balance, default, col="darkorange",
                   pch="|", cex=0.5, main="Regression",
                   xlab="Balance", ylab="Probability Default"))
abline(h=c(1, 0), lty=2)
abline(model1, col="dodgerblue", lwd=2)


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")
```

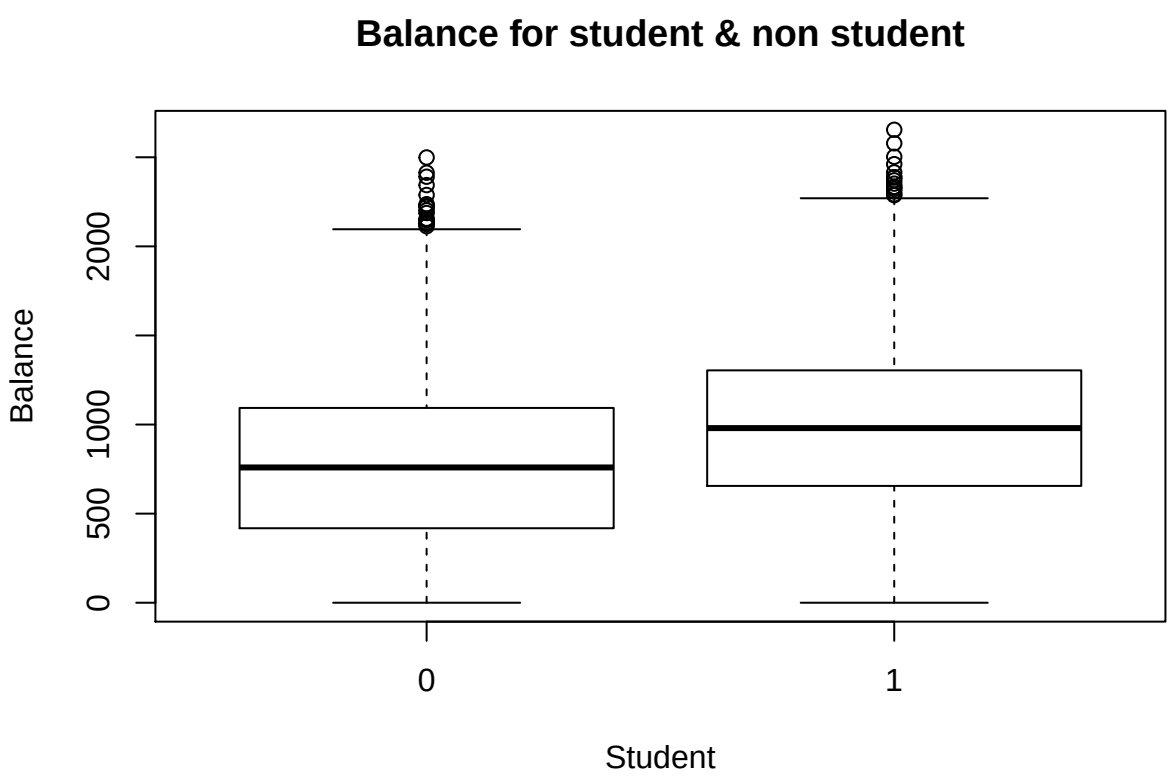



```
#balance
model2 <- glm(default ~ balance, data=Default, family=binomial)
#student
model3 <- glm(default ~ student, data=Default, family=binomial)
#all
model4 <- glm(default ~ balance + income + student, data=Default, family=binomial)
```

```
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")
```



5. Multinomial Logistic Regression

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

$$p_1 = p[y = 1] = \frac{1}{1 + \sum_{i=2}^{g} e^{\alpha_i + x\beta_i}} \tag{6}$$

$$p_k = p[y = k] = \frac{e^{\alpha_k + x\beta_k}}{1 + \sum_{i=2}^{g} e^{\alpha_i + x\beta_i}} fork = 2, ..., g \tag{7}$$

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

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

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 $\beta_k = 1.5$, 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

$$\frac{p_k}{p_r} = \frac{e^{\alpha_k + x\beta_k}}{e^{\alpha_r + x\beta_r}} = e^{\alpha k - \alpha r + x(\beta_k - \beta r)} \tag{8}$$

$$log(\frac{p_k}{p_1}) = \alpha_k - \alpha_r + x(\beta_k - \beta r) \tag{9}$$

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)
```

```
Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control
$wzepsilon): 1 elements replaced by 1.819e-12

Warning in checkwz(wz, M = M, trace = trace, wzepsilon = control
$wzepsilon): 1 elements replaced by 1.819e-12
```

```
summary(m_model)
```

```
Call:
vglm(formula = y ~ x, family = multinomial)

Pearson residuals:
                      Min      1Q    Median         3Q   Max
log(mu[,1]/mu[,5]) -21.99 -0.1517 -0.006299 -3.645e-05 1.374
log(mu[,2]/mu[,5]) -22.34 -0.4769 -0.126095 -1.508e-02 5.494
log(mu[,3]/mu[,5]) -22.16 -0.4613 -0.151788 -3.604e-02 3.252
log(mu[,4]/mu[,5]) -20.39 -0.3847 -0.116373 -2.463e-03 1.746

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept):1  17.7285     3.1770   5.580 2.40e-08 ***
(Intercept):2  14.3446     3.0229   4.745 2.08e-06 ***
(Intercept):3  12.8212     2.9442   4.355 1.33e-05 ***
(Intercept):4   6.3487     2.3423   2.710  0.00672 **
x:1            -5.9260     0.9920  -5.974 2.32e-09 ***
x:2            -3.8748     0.7796  -4.970 6.68e-07 ***
x:3            -3.2663     0.7269  -4.493 7.01e-06 ***
x:4            -1.3875     0.5102  -2.720  0.00654 **
---
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: 190.0822 on 392 degrees of freedom

Log-likelihood: -95.0411 on 392 degrees of freedom

Number of iterations: 7
```

```r
# Probability in each group
head(round(fitted(m_model), 2))
```

```
     1    2    3 4 5
1 0.60 0.27 0.13 0 0
2 0.91 0.07 0.02 0 0
3 0.51 0.32 0.17 0 0
4 0.96 0.04 0.01 0 0
5 0.93 0.06 0.01 0 0
6 0.99 0.01 0.00 0 0
```

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

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

```
    predict
true  1  2  3  4  5
   1 18  2  0  0  0
   2  5  8  3  4  0
   3  2  5  9  4  0
   4  0  0  3 13  4
   5  0  1  0  5 14
```