# Econometrics II TA Session #3

Hiroki Kato

## 1 Empirical Application of Binary Model: Titanic Survivors

**Brief Background**. "Women and children first" is a behavioral norm, which women and children are saved first in a life-threatening situation. This code was made famous by the sinking of the Titanic in 1912. An empirical application investigates characteristics of survivors of Titanic to answer whether crews obeyed the code or not.

**Data**. We use an open data about Titanic survivors [1]. Although this dataset contains many variables, we use only four variables: `survived`, `age`, `fare`, and `sex`. We summarize descpritons of variables as follows:

- `survived`: a binary variable taking 1 if a passenger survived.
- `age`: a continuous variable representing passeger's age.
- `fare`: a continuous variable representing how much passeger paid.
- `sex`: a string variable representing passenger's sex.

Using `sex`, we will make a binary variable, called `female`, taking 1 if passeger is female. Intead of `sex`, we use `female` variable in regression.

```
dt <- read.csv(
  file = "./data/titanic.csv",
  header = TRUE,  sep = ",", row.names = NULL,  stringsAsFactors = FALSE)

dt$female <- ifelse(dt$sex == "female", 1, 0)
dt <- subset(dt, !is.na(survived)&!is.na(age)&!is.na(fare)&!is.na(female))

dt <- dt[,c("survived", "age", "fare", "female")]
head(dt)
```

```
##   survived   age     fare female
## 1        1 29.00 211.3375      1
## 2        1  0.92 151.5500      0
## 3        0  2.00 151.5500      1
## 4        0 30.00 151.5500      0
```

[1] data source: http://biostat.mc.vanderbilt.edu/DataSets.

```
## 5          0 25.00 151.5500       1
## 6          1 48.00  26.5500       0
```

**Model**. In a binary model, a dependent (outcome) variable $y_i$ takes only two values, i.e., $y_i \in \{0, 1\}$. A binary variable is sometimes called a *dummy* variable. In this application, the outcome variable is `survived`. Explanatory variables are `female`, `age`, and `fare`. The regression function is

$$
\begin{aligned}
&\mathbb{E}[survived|female, age, fare] \\
=&\mathbb{P}[survived = 1|female, age, fare] = G(\beta_0 + \beta_1 female + \beta_2 age + \beta 3 fare).
\end{aligned}
$$

The function $G(\cdot)$ is arbitrary function. In practice, we often use following three specifications:

- Linear probability model (LPM): $G(\mathbf{x}_i\beta) = \mathbf{x}_i\beta$.
- Probit model: $G(\mathbf{x}_i\beta) = \Phi(\mathbf{x}_i\beta)$ where $\Phi(\cdot)$ is the standard Gaussian cumulative function.
- Logit model: $G(\mathbf{x}_i\beta) = 1/(1 + \exp(-\mathbf{x}_i\beta))$.

## 1.1 Linear Probability Model

The linear probability model specifys that $G(a)$ is linear in $a$, that is,

$$
\mathbb{P}[survived = 1|female, age, fare] = \beta_0 + \beta_1 female + \beta_2 age + \beta 3 fare.
$$

This model can be estimated using the OLS method. In `R`, we can use the OLS method, running `lm()` function.

```
model <- survived ~ female + age + fare
LPM <- lm(model, data = dt)
```

However, `lm()` function does not deal with heteroskedasticity problem. To resolve it, we need to claculate heteroskedasticity-robust standard errors using the White method.

$$
\hat{V}(\hat{\beta}) = \left( \frac{1}{n} \sum_i \mathbf{x}_i' \mathbf{x}_i \right)^{-1} \left( \frac{1}{n} \sum_i \hat{u}_i^2 \mathbf{x}_i' \mathbf{x}_i \right) \left( \frac{1}{n} \sum_i \mathbf{x}_i' \mathbf{x}_i \right)^{-1}
$$

```
# heteroskedasticity-robust standard errors
dt$"(Intercept)" <- 1
X <- as.matrix(dt[,c("(Intercept)", "female", "age", "fare")])
u <- diag(LPM$residuals^2)

XX <- t(X) %*% X
avgXX <- XX * nrow(X)^{-1}
inv_avgXX <- solve(avgXX)

uXX <- t(X) %*% u %*% X
```

```r
avguXX <- uXX * nrow(X)^{-1}

vcov_b <- (inv_avgXX %*% avguXX %*% inv_avgXX) * nrow(X)^{-1}
rse_b <- sqrt(diag(vcov_b))

# homoskedasticity-based standard errors
se_b <- sqrt(diag(vcov(LPM)))

print("The Variance of OLS"); vcov(LPM)
```

```
## [1] "The Variance of OLS"

##               (Intercept)        female           age          fare
## (Intercept)  9.754357e-04 -2.891381e-04 -2.333963e-05 -3.329763e-07
## female      -2.891381e-04  7.136865e-04  2.373259e-06 -1.272800e-06
## age         -2.333963e-05  2.373259e-06  8.026024e-07 -4.090649e-08
## fare        -3.329763e-07 -1.272800e-06 -4.090649e-08  5.524412e-08
```

```r
print("The Robust variance of OLS"); vcov_b
```

```
## [1] "The Robust variance of OLS"

##               (Intercept)        female           age          fare
## (Intercept)  1.133289e-03 -2.798532e-04 -2.789675e-05  2.813843e-07
## female      -2.798532e-04  7.903766e-04  3.169092e-06 -2.401923e-06
## age         -2.789675e-05  3.169092e-06  8.857523e-07 -3.650375e-08
## fare         2.813843e-07 -2.401923e-06 -3.650375e-08  4.071639e-08
```

```r
print("The Robust se using White method"); rse_b
```

```
## [1] "The Robust se using White method"

##  (Intercept)       female          age         fare
## 0.0336643606 0.0281136372 0.0009411442 0.0002017830
```

```r
print("The Robust t-value using White method"); coef(LPM)/rse_b
```

```
## [1] "The Robust t-value using White method"

## (Intercept)      female         age        fare
##    6.482874   18.229508   -1.884168    7.162302
```

Using the package `lmtest` and `sandwich` is the easiest way to calculate heteroskedasticity-robust standard errors and $t$-statistics.

```r
library(lmtest) #use function `coeftest`
library(sandwich) #use function `vcovHC`
coeftest(LPM, vcov = vcovHC(LPM, type = "HC0"))[, "Std. Error"]
```

```
##  (Intercept)       female          age         fare
```

```
## 0.0336643606 0.0281136372 0.0009411442 0.0002017830
```

```r
coeftest(LPM, vcov = vcovHC(LPM, type = "HC0"))[, "t value"]
```

```
## (Intercept)      female         age        fare
##    6.482874   18.229508   -1.884168    7.162302
```

Finally, we summarize results of linear probability model in table 1. We will discuss interpretation of results and goodness-of-fit of LPM later.

```r
# t-stats
t_b <- coef(LPM)/se_b
rt_b <- coef(LPM)/rse_b
# p-value Pr( > |t|)
p_b <- pt(abs(t_b), df = nrow(X)-ncol(X), lower = FALSE)*2
rp_b <- pt(abs(rt_b), df = nrow(X)-ncol(X), lower = FALSE)*2

library(stargazer)
stargazer(
  LPM, LPM,
  se = list(se_b, rse_b), t = list(t_b, rt_b), p = list(p_b, rp_b),
  t.auto = FALSE, p.auto = FALSE,
  report = "vcstp", keep.stat = c("n"),
  add.lines = list(
    c("Standard errors", "Homoskedasticity-based", "Heteroskedasticity-robust")),
  title = "Results of Linear Probability Model", label = "LPM",
  type = "latex", header = FALSE, font.size = "small",
  omit.table.layout = "n", table.placement = "h"
)
```

## 1.2 Probit and Logit Model

Unlike LPM, the probit and logit model must be estimated using the ML method. The probability of observing $y_i$ is

$$p_\beta(y_i|\mathbf{x}_i) = \mathbb{P}(y_i = 1|x_i)^{y_i}[1 - \mathbb{P}(y_i = 1|x_i)]^{1-y_i} = G(\mathbf{x}_i\beta)^{y_i}(1 - G(\mathbf{x}_i\beta))^{1-y_i}.$$

Taking logalithm yields

$$\log p_\beta(y_i|\mathbf{x}_i) = y_i \log(G(\mathbf{x}_i\beta)) + (1 - y_i)\log(1 - G(\mathbf{x}_i\beta)).$$

The log-likelihood is

$$M_n(\beta) = \sum_{i=1}^n \log p_\beta(y_i|\mathbf{x}_i).$$

The MLE $\hat{\beta}$ holds that the score, which is the first-order derivatives with respect to $\beta$, is equal to 0. That is $\nabla_\beta M_n(\hat{\beta}) = 0$. For both logit and probit model, the Hessian matrix,

Table 1: Results of Linear Probability Model

| | *Dependent variable:* | |
|---|---|---|
| | survived | |
| | (1) | (2) |
| female | 0.512 | 0.512 |
| | (0.027) | (0.028) |
| | t = 19.184 | t = 18.230 |
| | p = 0.000 | p = 0.000 |
| age | −0.002 | −0.002 |
| | (0.001) | (0.001) |
| | t = −1.979 | t = −1.884 |
| | p = 0.049 | p = 0.060 |
| fare | 0.001 | 0.001 |
| | (0.0002) | (0.0002) |
| | t = 6.149 | t = 7.162 |
| | p = 0.000 | p = 0.000 |
| Constant | 0.218 | 0.218 |
| | (0.031) | (0.034) |
| | t = 6.988 | t = 6.483 |
| | p = 0.000 | p = 0.000 |
| Standard errors | Homoskedasticity-based | Heteroskedasticity-robust |
| Observations | 1,045 | 1,045 |

$\nabla^2_{\beta\beta'} M_n(\beta)$, is always negative definite. This implies that log-likelihood function based on both models is grobally concave, and ensures that the MLE maximizes the log-likelihood function. The first-order condition of the probit model is

$$\nabla_\beta M_n(\hat{\beta}) = \sum_{i=1}^n \left( y_i - \Phi(\mathbf{x}_i \hat{\beta}) \right) \frac{\phi(\mathbf{x}_i \hat{\beta})}{\Phi(\mathbf{x}_i \hat{\beta})(1 - \phi(\mathbf{x}_i \hat{\beta}))} = 0.$$

The first-order condition of the logit model is

$$\nabla_\beta M_n(\hat{\beta}) = \sum_{i=1}^n \left( y_i - G(\mathbf{x}_i \hat{\beta}) \right) \mathbf{x}_i' = 0.$$

Since it is hard for us to solve this condition analytically, we obtain estimators using numerical procedure.

In `R`, the function `nlm()` provides the Newton-Raphson algorithm to minimize the function

[2]. To run this function, we need to define the log-likelihood function (`LogLik`) beforehand. Moreover, since we need to give initial values in augments, we use coefficients estimated by OLS. Alternatively, we often use `glm()` function. Using this function, we do not need to define the log-likelihood function and initial values. Since estimates of `glm()` are approximate to estiamtes of `nlm()`, we can use this function fairly. In this application, we use `nlm` function to minimize the log-likelihood function.

```r
Y <- dt$survived
female <- dt$female; age <- dt$age; fare <- dt$fare

# log-likelihood
LnLik <- function(b, model = c("probit", "logit")) {

  xb <- b[1]+ b[2]*female + b[3]*age + b[4]*fare

  if (model == "probit") {
    L <- pnorm(xb)
  } else {
    L <- 1/(1 + exp(-xb))
  }

  LL_i <- Y * log(L) + (1 - Y) * log(1 - L)
  LL <- -sum(LL_i)

  return(LL)
}

#Newton-Raphson
init <- c(0.218, 0.512, -0.002, 0.001)
probit <- nlm(LnLik, init, model = "probit", hessian = TRUE)

label <- c("(Intercept)", "female", "age", "fare")
names(probit$estimate) <- label
colnames(probit$hessian) <- label; rownames(probit$hessian) <- label

b_probit <- probit$estimate
vcov_probit <- solve(probit$hessian); se_probit <- sqrt(diag(vcov_probit))
LL_probit <- -probit$minimum

#glm function
model <- survived ~ female + age + fare
probit_glm <- glm(model, data = dt, family = binomial("probit"))
```

[2] `optim()` function is an another way to minimize the function. Especially, the function `optim(method = "BFGS")` provides the Quasi-Newton algorithm which carries on the spirit of Newton method.

```r
#result
print("The MLE of probit model using nlm"); b_probit
```

```
## [1] "The MLE of probit model using nlm"

##  (Intercept)       female          age         fare
## -0.813995120  1.435384017 -0.006415761  0.005954843
```

```r
print("The Variance of probit model using nlm"); vcov_probit
```

```
## [1] "The Variance of probit model using nlm"

##               (Intercept)        female           age          fare
## (Intercept)  1.149118e-02 -3.569149e-03 -2.654781e-04 -1.375309e-05
## female      -3.569149e-03  8.251773e-03  2.000500e-05 -5.991997e-06
## age         -2.654781e-04  2.000500e-05  9.630856e-06 -6.874343e-07
## fare        -1.375309e-05 -5.991997e-06 -6.874343e-07  1.103772e-06
```

```r
print("The se of probit model using nlm"); se_probit
```

```
## [1] "The se of probit model using nlm"

## (Intercept)      female         age        fare
## 0.107196925 0.090839272 0.003103362 0.001050606
```

```r
print("The coefficients of probit using glm"); coef(probit_glm)
```

```
## [1] "The coefficients of probit using glm"

##  (Intercept)       female          age         fare
## -0.814075240  1.435384903 -0.006413717  0.005955479
```

```r
print("The se of probit using glm"); sqrt(diag(vcov(probit_glm)))
```

```
## [1] "The se of probit using glm"

## (Intercept)      female         age        fare
## 0.108614928 0.090860818 0.003139413 0.001056285
```

Using `LogLik`, we can also estimate logit model by Newton-Raphson algorithm. To compare result, we also use `glm()` function.

```r
#Newton-Raphson
logit <- nlm(LnLik, init, model = "logit", hessian = TRUE)

names(logit$estimate) <- label
colnames(logit$hessian) <- label; rownames(logit$hessian) <- label

b_logit <- logit$estimate
vcov_logit <- solve(logit$hessian); se_logit <- sqrt(diag(vcov_logit))
LL_logit <- -logit$minimum
```

```r
#glm function
logit_glm <- glm(model, data = dt, family = binomial("logit"))

#result
print("The MLE of logit model"); b_logit
```

```
## [1] "The MLE of logit model"

## (Intercept)      female         age        fare
## -1.33719278  2.35516448 -0.01105760  0.01002878
```

```r
print("The Variance of logit model"); vcov_logit
```

```
## [1] "The Variance of logit model"

##               (Intercept)        female           age          fare
## (Intercept)  0.0351392692 -1.052616e-02 -8.031155e-04 -4.682750e-05
## female      -0.0105261593  2.411636e-02  3.401375e-05 -7.818252e-06
## age         -0.0008031155  3.401375e-05  2.939124e-05 -2.170680e-06
## fare        -0.0000468275 -7.818252e-06 -2.170680e-06  3.448283e-06
```

```r
print("The se of logit model"); se_logit
```

```
## [1] "The se of logit model"

## (Intercept)      female         age        fare
## 0.187454712 0.155294438 0.005421369 0.001856955
```

```r
print("The coefficients of logit using glm"); coef(logit_glm)
```

```
## [1] "The coefficients of logit using glm"

## (Intercept)      female         age        fare
## -1.33727469  2.35516632 -0.01105553  0.01002942
```

```r
print("The se of logit using glm"); sqrt(diag(vcov(logit_glm)))
```

```
## [1] "The se of logit using glm"

## (Intercept)      female         age        fare
## 0.187350369 0.155280058 0.005424281 0.001847912
```

As a result, table 2 summarizes results of probit model and logit model. $t$-statistics represents $z$-value which follows the standard normal distribution. Standard errors are in parentheses. We will discuss interpretation of results and goodness-of-fit later.

```r
# z-value
z_probit <- b_probit/se_probit
z_logit <- b_logit/se_logit
```

```r
# Pr(>|z|)
p_probit <- pnorm(abs(z_probit), lower = FALSE)*2
p_logit <- pnorm(abs(z_logit), lower = FALSE)*2

stargazer(
  probit_glm, logit_glm,
  coef = list(b_probit, b_logit), se = list(se_probit, se_logit),
  t = list(z_probit, z_logit), p = list(p_probit, p_logit),
  t.auto = FALSE, p.auto = FALSE,
  report = "vcstp", keep.stat = c("n"),
  add.lines = list(
    c("Log-Likelihood", round(LL_probit, 3), round(LL_logit, 3))),
  title = "Results of Probit and Logit model",
  label = "probit_logit",
  type = "latex", header = FALSE, font.size = "small",
  table.placement = "h", omit.table.layout = "n"
)
```

Table 2: Results of Probit and Logit model

| | *Dependent variable:* | |
|---|---|---|
| | survived | |
| | *probit* | *logistic* |
| | (1) | (2) |
| female | 1.435 | 2.355 |
| | (0.091) | (0.155) |
| | t = 15.801 | t = 15.166 |
| | p = 0.000 | p = 0.000 |
| age | −0.006 | −0.011 |
| | (0.003) | (0.005) |
| | t = −2.067 | t = −2.040 |
| | p = 0.039 | p = 0.042 |
| fare | 0.006 | 0.010 |
| | (0.001) | (0.002) |
| | t = 5.668 | t = 5.401 |
| | p = 0.000 | p = 0.00000 |
| Constant | −0.814 | −1.337 |
| | (0.107) | (0.187) |
| | t = −7.593 | t = −7.133 |
| | p = 0.000 | p = 0.000 |
| Log-Likelihood | -530.404 | -530.947 |
| Observations | 1,045 | 1,045 |