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

$$\begin{aligned} & E[survived|female, age, fare] \\ = & \mathbb{P}[survived = 1|female, age, fare] = G(\beta_0 + \beta_1 female + \beta_2 age + \beta_3 fare). \end{aligned}$$

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 ~ factor(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))

label <- c("(Intercept)", "factor(female)1", "age", "fare")
names(rse_b) <- label

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

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

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

##                   (Intercept) factor(female)1           age          fare
## (Intercept)      9.754357e-04   -2.891381e-04 -2.333963e-05 -3.329763e-07
## factor(female)1 -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) factor(female)1             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) factor(female)1             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) factor(female)1             age            fare
##    0.0336643606    0.0281136372    0.0009411442    0.0002017830
```

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

```
##     (Intercept) factor(female)1             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"),
  covariate.labels = c("Female = 1"),
  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

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

Table 1: Results of Linear Probability Model

	Dependent variable:	
	survived	
	(1)	(2)
Female = 1	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

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 β, is equal to 0. That is $\nabla_\beta M_n(\hat{\beta}) = 0$. For both logit and probit model, the Hessian matrix, $\nabla^2_{\beta\beta'} M_n(\beta)$, 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.

The asymptotic distribution of $\hat{\beta}$ is $\hat{\beta} \xrightarrow{d} N(\beta, \Sigma_\beta)$ where

$$\Sigma_\beta = -\left(\sum_i E[E[\nabla^2_{\beta\beta'} \log p_\beta(y_i|\mathbf{x}_i)|\mathbf{x}_i]] \right)^{-1}.$$

In practice, we replace $E[E[\nabla^2_{\beta\beta'} \log p_\beta(y_i|\mathbf{x}_i)|\mathbf{x}_i]]$ by

$$\frac{1}{n}\sum_i \nabla^2_{\beta\beta'} \log p_{\hat{\beta}}(y_i|\mathbf{x}_i),$$

that is,

$$\hat{\Sigma}_\beta = \left(\sum_i \nabla^2_{\beta\beta'}(-\log p_{\hat{\beta}}(y_i|\mathbf{x}_i)) \right)^{-1}.$$

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 (`LnLik`) 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 command safely. In this application, we use `nlm` function to minimize the log-likelihood function.

```
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)
```

[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
  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)", "factor(female)1", "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 ~ factor(female) + age + fare
probit_glm <- glm(model, data = dt, family = binomial("probit"))

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

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

##     (Intercept) factor(female)1             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) factor(female)1           age          fare
## (Intercept)      1.149118e-02   -3.569149e-03 -2.654781e-04 -1.375309e-05
## factor(female)1 -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) factor(female)1             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) factor(female)1             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) factor(female)1             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)

label <- c("(Intercept)", "factor(female)1", "age", "fare")
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

#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) factor(female)1             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) factor(female)1           age          fare
## (Intercept)      0.0351392692   -1.052616e-02 -8.031155e-04 -4.682750e-05
## factor(female)1 -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) factor(female)1             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) factor(female)1             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) factor(female)1             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

# 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"),
  covariate.labels = c("Female = 1"),
  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"
)
```

1.3 Interpretaions

In the linear probability model, interepretations of coefficients are straight-forward. The coefficient β_1 is the change in survival probability given a one-unit increase in continuous variable x. In the case of discrete variable, the coefficient β_1 is the difference in survival

Table 2: Results of Probit and Logit model

	Dependent variable:	
	survived	
	probit	*logistic*
	(1)	(2)
Female = 1	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

probability between two groups. However, when we use the probit or logit model, it is hard for us to interepret results because the partial effect is not constant across other covariates. As an illustration, the partial effect of continuous variable `age` is

$$\partial_{age}\mathbb{P}[survived = 1|female, age, fare] = \begin{cases} \beta_2 & \text{if LPM} \\ \phi(\mathbf{x}_i\beta)\beta_2 & \text{if Probit} \\ \frac{\exp(-\mathbf{x}_i\beta)}{(1+\exp(-\mathbf{x}_i\beta))^2}\beta_2 & \text{if Logit} \end{cases}.$$

The partial effect of dummy variable `female` is

$$\begin{aligned} &\mathbb{P}[survived = 1|female = 1, age, fare] - \mathbb{P}[survived = 1|female = 0, age, fare] \\ =& \begin{cases} \beta_1 & \text{if LPM} \\ \Phi(\beta_0 + \beta_1 + \beta_2 age + \beta_3 fare) - \Phi(\beta_0 + \beta_1 + \beta_2 age + \beta_3 fare) & \text{if Probit} \\ \Lambda(\beta_0 + \beta_1 + \beta_2 age + \beta_3 fare) - \Lambda(\beta_0 + \beta_1 + \beta_2 age + \beta_3 fare) & \text{if Logit} \end{cases}, \end{aligned}$$

where $\Lambda(a) = 1/(1+\exp(-a))$.

The first solution is to compute the partial effect at interesting values of $\mathbf{x}_i$. We often use the sample average of covariates ("average" person) to plugin in the partial effect formula. This is sometimes called *marginal effect at means*. However, since it is unclear what the sample average of dummy variable represents, the marginal effect at means may be hard to explain.

The second solution is to compute the average value of partial effect across the population, that is,

$$\partial_{x_{ij}} \mathbb{P}[y_i = 1 | \mathbf{x}_i] = \beta_j E[g(\mathbf{x}_i \beta)],$$

or, in the case of discrete variable,

$$E[\mathbb{P}[y_i = 1 | x_{ij} = 1, \mathbf{x}_{i,-k}] - \mathbb{P}[y_i = 1 | x_{ij} = 0, \mathbf{x}_{i,-k}]].$$

This is called *average marginal effect* (AME). When we use dummy variables as explanatory variables, we should use this solution.

Standard errors of average marginal effect can be obtained by the Delta method. Let $h_{ij}(\hat{\beta})$ be marginal (partial) effect of the variable x_j for unit i. Then, AME is $h_j(\hat{\beta}) = E[h_{ij}(\hat{\beta})]$. The Delta method implies that $h_j(\hat{\beta}) \xrightarrow{d} N(h_j(\beta), \nabla_\beta h_j(\hat{\beta}) V(\beta) (\nabla_\beta h_j(\hat{\beta}))')$, where V is variance of β, and

$$\nabla_\beta h_j(\hat{\beta}) = \begin{pmatrix} \frac{\partial h_j(\hat{\beta})}{\partial \beta_1} & \dots & \frac{\partial h_j(\hat{\beta})}{\partial \beta_k} \end{pmatrix}$$

When you use the `nlm` function to obtaine MLE, we need to calculate standard errors manually. The `DeltaAME` function is a function returing average marginal effect and its standard errors.

```
DeltaAME <- function(b, X, vcov, jbin = NULL, model = c("probit", "logit")) {
  Xb <- numeric(nrow(X))
  for (i in 1:length(b)) {
     Xb <- Xb + b[i] * X[,i]
  }

  if (model == "probit") {
    dens <- dnorm(Xb)
    grad <- -Xb * dens
  } else {
    dens <- exp(-Xb)/(1 + exp(-Xb))^2
    grad <- dens * (-1+2*exp(-Xb)/(1+exp(-Xb)))
  }

  ame <- mean(dens) * b
  if (!is.null(jbin)) {
    for (i in jbin) {
```

```
    val1 <- X[,-i] %*% matrix(b[-i], ncol = 1) + b[i]
    val0 <- X[,-i] %*% matrix(b[-i], ncol = 1)
    if (model == "probit") {
      amed <- mean(pnorm(val1) - pnorm(val0))
    } else {
      amed <- mean((1/(1 + exp(-val1))) - (1/(1 + exp(-val0))))
    }
    ame[i] <- amed
  }
}

e <- NULL
for (i in 1:length(b)) {
  e <- c(e, rep(mean(X[,i] * grad), length(b)))
}

Jacob <- matrix(e, nrow = length(b), ncol = length(b))

for (i in 1:nrow(Jacob)) {
  Jacob[i,] <- b[i] * Jacob[i,]
}
diag(Jacob) <- diag(Jacob) + rep(mean(dens), length(b))

if (!is.null(jbin)) {
  for (i in jbin) {
    val1 <- X[,-i] %*% matrix(b[-i], ncol = 1) + b[i]
    val0 <- X[,-i] %*% matrix(b[-i], ncol = 1)
    de <- NULL
    if (model == "probit") {
      for (j in 1:length(b)) {
         if (j != i) {
          dep <- X[,j] * (dnorm(val1) - dnorm(val0))
          de <- c(de, mean(dep))
         } else {
          dep <- dnorm(val1)
          de <- c(de, mean(dep))
         }
      }
    } else {
      for (j in 1:length(b)) {
         if (j != i) {
          dep <- X[,j] *
            ((exp(-val1)/(1 + exp(-val1))^2) - (exp(-val0)/(1 + exp(-val0))^2))
          de <- c(de, mean(dep))
```

```r
          } else {
           dep <- exp(-val1)/(1 + exp(-val1))^2
           de <- c(de, mean(dep))
          }
       }
     }
     Jacob[i,] <- de
   }
  }

  label <- names(b)
  colnames(Jacob) <- label; rownames(Jacob) <- label

  vcov_ame <- Jacob %*% vcov %*% t(Jacob)
  se_ame <- sqrt(diag(vcov_ame))
  z_ame <- ame/se_ame
  p_ame <- pnorm(abs(z_ame), lower = FALSE)*2

  return(list(AME = ame[-1], SE = se_ame[-1], zval = z_ame[-1], pval = p_ame[-1]))
}

X <- as.matrix(dt[,c("(Intercept)", "female", "age", "fare")])
ame_probit <- DeltaAME(b_probit, X, vcov_probit, jbin = 2, model = "probit")
ame_logit <- DeltaAME(b_logit, X, vcov_logit, jbin = 2, model = "logit")

print("AME of probit estimates"); ame_probit$AME
```

```
## [1] "AME of probit estimates"

## factor(female)1             age            fare
##     0.508541457    -0.001824620     0.001693537
```

```r
print("AME of logit estimates"); ame_logit$AME
```

```
## [1] "AME of logit estimates"

## factor(female)1             age            fare
##     0.507384641    -0.001823282     0.001653639
```

```r
print("SE of AME of probit estimates"); ame_probit$SE
```

```
## [1] "SE of AME of probit estimates"

## factor(female)1             age            fare
##    0.0285474135    0.0008786651    0.0002874017
```

```r
print("SE of AME of logit estimates"); ame_logit$SE
```

```
## [1] "SE of AME of logit estimates"
```

```
## factor(female)1             age            fare
##    0.0287277842    0.0008897546    0.0002948759
```

When we use the `glm` function, we can use the function `margins` in the library `margins` to obtain the average marginal effect.

```r
library(margins)
summary(margins(probit_glm))
```

```
##   factor     AME     SE       z      p   lower   upper
##      age -0.0018 0.0009 -2.0520 0.0402 -0.0036 -0.0001
##     fare  0.0017 0.0003  5.8398 0.0000  0.0011  0.0023
##  female1  0.5085 0.0286 17.7611 0.0000  0.4524  0.5647
```

```r
summary(margins(logit_glm))
```

```
##   factor     AME     SE       z      p   lower   upper
##      age -0.0018 0.0009 -2.0480 0.0406 -0.0036 -0.0001
##     fare  0.0017 0.0003  5.6338 0.0000  0.0011  0.0022
##  female1  0.5074 0.0287 17.6652 0.0000  0.4511  0.5637
```

Table 3 shows results of linear probability model, probit model, and logit model. In the probit and logit model, coefficients report average marginal effects, t-statistics report z-statistics which follows the standard normal distribution.

All specifications shows that the survival probability of female is about 50% point higher than of male, which is statistically significant. Moreover, the survival probability is decreasing in age, which implies children are more likely to survive. However, the size of coefficient is small. Overall, crews obeyed the code of "women and children first", but the survival probability of children is not largely different from of adult.

1.4 Model Fitness

There are two measurements of goodness-of-fit. First, the *percent correctly predicted* reports the percentage of unit whose predicted y_i matches the actual y_i. The predicted y_i takes one if $G(\mathbf{x}_i\hat{\beta}) > 0.5$, and takes zero if $G(\mathbf{x}_i\hat{\beta}) \leq 0.5$.

```r
Y <- dt$survived
X <- as.matrix(dt[,c("(Intercept)", "female", "age", "fare")])

Xb_lpm <- X %*% matrix(coef(LPM), ncol = 1)
Xb_probit <- X %*% matrix(b_probit, ncol = 1)
Xb_logit <- X %*% matrix(b_logit, ncol = 1)

hatY_lpm <- ifelse(Xb_lpm > 0.5, 1, 0)
hatY_probit <- ifelse(pnorm(Xb_probit) > 0.5, 1, 0)
```

```r
hatY_logit <- ifelse(1/(1 + exp(-Xb_logit)) > 0.5, 1, 0)

pcp_lpm <- round(sum(Y == hatY_lpm)/nrow(X), 4)
pcp_probit <- round(sum(Y == hatY_probit)/nrow(X), 4)
pcp_logit <- round(sum(Y == hatY_logit)/nrow(X), 4)
```

Second measurement is the *pseudo R-squared*. The pseudo R-squared is obtained by $1 - \sum_i \hat{u}_i^2 / \sum_i y_i^2$, where $\hat{u}_i = y_i - G(\mathbf{x}_i \hat{\beta})$.

```r
Y2 <- Y^2

hatu_lpm <- (Y - Xb_lpm)^2
hatu_probit <- (Y - pnorm(Xb_probit))^2
hatu_logit <- (Y - 1/(1 + exp(-Xb_logit)))^2

pr2_lpm <- round(1 - sum(hatu_lpm)/sum(Y2), 4)
pr2_probit <- round(1 - sum(hatu_probit)/sum(Y2), 4)
pr2_logit <- round(1 - sum(hatu_logit)/sum(Y2), 4)
```

Table 3 summarizes two measurements of model fitness. There is little difference among LPM, probit model, and logit model.

```r
stargazer(
  LPM, probit_glm, logit_glm,
  coef = list(coef(LPM), ame_probit$AME, ame_logit$AME),
  se = list(rse_b, ame_probit$SE, ame_logit$SE),
  t = list(rt_b, ame_probit$zval, ame_logit$zval),
  p = list(rp_b, ame_probit$pval, ame_logit$pval),
  t.auto = FALSE, p.auto = FALSE,
  omit = c("Constant"), covariate.labels = c("Female = 1"),
  report = "vcstp", keep.stat = c("n"),
  add.lines = list(
    c("Percent correctly predicted", pcp_lpm, pcp_probit, pcp_logit),
    c("Pseudo R-squared", pr2_lpm, pr2_probit, pr2_logit)
  ),
  omit.table.layout = "n", table.placement = "t",
  title = "Titanic Survivors: LPM, Probit (AME), and Logit (AME)",
  label = "titanic",
  type = "latex", header = FALSE
)
```

Table 3: Titanic Survivors: LPM, Probit (AME), and Logit (AME)

	Dependent variable:		
	survived		
	OLS	*probit*	*logistic*
	(1)	(2)	(3)
Female = 1	0.512	0.509	0.507
	(0.028)	(0.029)	(0.029)
	t = 18.230	t = 17.814	t = 17.662
	p = 0.000	p = 0.000	p = 0.000
age	−0.002	−0.002	−0.002
	(0.001)	(0.001)	(0.001)
	t = −1.884	t = −2.077	t = −2.049
	p = 0.060	p = 0.038	p = 0.041
fare	0.001	0.002	0.002
	(0.0002)	(0.0003)	(0.0003)
	t = 7.162	t = 5.893	t = 5.608
	p = 0.000	p = 0.000	p = 0.00000
Percent correctly predicted	0.7799	0.7742	0.7742
Pseudo R-squared	0.5946	0.5945	0.594
Observations	1,045	1,045	1,045

2 Empirical Application of Ordered Probit and Logit Model: Housing as Status Goods

Breif Background. Social image may affect consumption behavior. Specifically, a desire to signal high income or wealth may cause consumers to purchase status goods. In this application, we explore whether living in an upper floor serves as a status goods.

Data. We use the housing data originally coming from the American Housing Survey conducted in 2013 [3]. We use the following variable

- `Level`: ordered value of a story of respondent's living (1:Low - 4:High)
- `Levelnum`: variable we recode the response `Level` as 25, 50, 75, 100. This represents the extent of floor height.
- `lnPrice`: logged price of housing (proxy for quality of house)
- `Top25`: a dummy variable taking one if household income is in the top 25 percentile in

[3] https://www.census.gov/programs-surveys/ahs.html. This is a repeated cross-section survey. We use the data at one time.

sample.

```r
house <- read.csv(file = "./data/housing.csv", header = TRUE,  sep = ",")
house <- house[,c("Level", "lnPrice", "Top25")]
house$Levelnum <- ifelse(
  house$Level == 1, 25,
  ifelse(house$Level == 2, 50,
  ifelse(house$Level == 3, 75, 100)))
head(house)
```

```
##   Level  lnPrice Top25 Levelnum
## 1     3 11.51294     0       75
## 2     4 11.51294     1      100
## 3     3 11.60824     0       75
## 4     3 11.69526     0       75
## 5     3 12.57764     0       75
## 6     3 12.64433     0       75
```

Model. The outcome variable is `Level` taking $\{1, 2, 3, 4\}$. Consider the following regression equation of a latent variable:

$$y_i^* = \mathbf{x}_i \beta + u_i,$$

where $\mathbf{x}_i = (lnPrice, Top25)$ and u_i is an error term. The relationship between the latent variable y_i^* and the observed outcome variable is

$$Level = \begin{cases} 1 & \text{if} \quad -\infty < y_i^* \leq a_1 \\ 2 & \text{if} \quad a_1 < y_i^* \leq a_2 \\ 3 & \text{if} \quad a_2 < y_i^* \leq a_3 \\ 4 & \text{if} \quad a_3 < y_i^* < +\infty \end{cases}.$$

Consider the probability of realization of y_i, that is,

$$\begin{aligned} \mathbb{P}(y_i = k|\mathbf{x}_i) &= \mathbb{P}(a_{k-1} - \mathbf{x}_i \beta < u_i \leq a_k - \mathbf{x}_i \beta | \mathbf{x}_i) \\ &= G(a_k - \mathbf{x}_i \beta) - G(a_{k-1} - \mathbf{x}_i \beta), \end{aligned}$$

where $a_4 = +\infty$ and $a_0 = -\infty$. Then, the likelihood function is defined by

$$p((y_i|\mathbf{x}_i), i = 1, \dots, n; \beta, a_1, \dots, a_3) = \prod_{i=1}^{n} \prod_{k=1}^{4} (G(a_k - \mathbf{x}_i \beta) - G(a_{k-1} - \mathbf{x}_i \beta))^{I_{ik}}.$$

where I_{ik} is a indicator variable taking 1 if $y_i = k$. Finally, the log-likelihood function is

$$M(\beta, a_1, a_2, a_3) = \sum_{i=1}^{n} \sum_{k=1}^{4} I_{ik} \log(G(a_k - \mathbf{x}_i \beta) - G(a_{k-1} - \mathbf{x}_i \beta)).$$

Usually, $G(a)$ assumes the standard normal distribution, $\Phi(a)$, or the logistic distribution, $1/(1 + \exp(-a))$.

In R, the library (package) MASS provides the polr function which estimates the ordered probit and logit model. Although we can use the nlm function when we define the log-likelihood function, we do not report this method. To compare results, we use the variable Levelnum as outcome variable, and apply the linear regression model.

```r
library(MASS)
library(tidyverse) #use case_when()

ols <- lm(Levelnum ~ lnPrice + Top25, data = house)

model <- factor(Level) ~ lnPrice + Top25
oprobit <- polr(model, data = house, method = "probit")
ologit <- polr(model, data = house, method = "logistic")

a_oprobit <- round(oprobit$zeta, 3)
a_ologit <- round(ologit$zeta, 3)

xb_oprobit <- oprobit$lp
xb_ologit <- ologit$lp

hatY_oprobit <- case_when(
  xb_oprobit <= oprobit$zeta[1] ~ 1,
  xb_oprobit <= oprobit$zeta[2] ~ 2,
  xb_oprobit <= oprobit$zeta[3] ~ 3,
  TRUE ~ 4
)
hatY_ologit <- case_when(
  xb_ologit <= ologit$zeta[1] ~ 1,
  xb_ologit <= ologit$zeta[2] ~ 2,
  xb_ologit <= ologit$zeta[3] ~ 3,
  TRUE ~ 4
)

pred_oprobit <- round(sum(house$Level == hatY_oprobit)/nrow(house), 3)
pred_ologit <- round(sum(house$Level == hatY_ologit)/nrow(house), 3)
```

2.1 Interepretations

Table 4 shows results. OLS model shows that respondents whose household income is in the top 25 percentile live in 3.7% higher floor than other respondents. This implies that high earners want to live in higher floor, which may serve as a status goods. The ordered probit and logit model are in line with this result. To evaluate two models quantitatively, consider the following equation.

$$E[Levelnum|\mathbf{x}_i] = 25\mathbb{P}[level = 1|\mathbf{x}_i] + 50\mathbb{P}[level = 2|\mathbf{x}_i] + 75\mathbb{P}[level = 3|\mathbf{x}_i] + 100\mathbb{P}[level = 4|\mathbf{x}_i].$$

We compute this equation with $Top25 = 1$ and $Top25 = 0$ at mean value of $lnPrice$ and take difference.

```
quantef <- function(model) {
  b <- coef(model)
  val1 <- mean(house$lnPrice)*b[1] + b[2]
  val0 <- mean(house$lnPrice)*b[1]

  prob <- matrix(c(rep(val1, 3), rep(val0, 3)), ncol = 2, nrow = 3)
  for (i in 1:3) {
    for (j in 1:2) {
      prob[i,j] <- pnorm(model$zeta[i] - prob[i,j])
    }
  }
  Ey1 <- 25*prob[1,1] + 50*(prob[2,1]-prob[1,1]) +
    75*(prob[3,1]-prob[2,1]) + 100*(1-prob[3,1])
  Ey0 <- 25*prob[1,2] + 50*(prob[2,2]-prob[1,2]) +
    75*(prob[3,2]-prob[2,2]) + 100*(1-prob[3,2])

  return(Ey1 - Ey0)
}

ef_oprobit <- round(quantef(oprobit), 3)
ef_ologit <- round(quantef(ologit), 3)
```

As a result, we obtain similar values to OLSE. In the ordered probit model, earners in the top 25 percentile live in 4.2% higher floor than others. In the ordered logit model, earners in the top 25 percentile live in 5.9% higher floor than others. Note that, in this application, model fitness seems to be bad because the percent correctly predicted is low.

```
stargazer(
  ols, oprobit, ologit,
  report = "vcstp", keep.stat = c("n"),
  omit = c("Constant"),
  add.lines = list(
    c("Cutoff value at 1|2", "", a_oprobit[1], a_ologit[1]),
    c("Cutoff value at 2|3", "", a_oprobit[2], a_ologit[2]),
    c("Cutoff value at 3|4", "", a_oprobit[3], a_ologit[3]),
    c("Quantitative Effect of Top25", "", ef_oprobit, ef_ologit),
    c("Percent correctly predicted", "", pred_oprobit, pred_ologit)
  ),
  omit.table.layout = "n", table.placement = "t",
  title = "Floor Level of House: Ordered Probit and Logit Model",
  label = "housing",
  type = "latex", header = FALSE
)
```

Table 4: Floor Level of House: Ordered Probit and Logit Model

	Dependent variable:		
	Levelnum	Level	
	OLS	*ordered probit*	*ordered logistic*
	(1)	(2)	(3)
lnPrice	0.348	0.012	0.019
	(0.430)	(0.016)	(0.026)
	t = 0.810	t = 0.777	t = 0.745
	p = 0.418	p = 0.438	p = 0.457
Top25	3.714	0.156	0.239
	(1.723)	(0.064)	(0.106)
	t = 2.156	t = 2.426	t = 2.259
	p = 0.032	p = 0.016	p = 0.024
Cutoff value at 1|2		-0.149	-0.25
Cutoff value at 2|3		0.246	0.384
Cutoff value at 3|4		0.97	1.574
Quantitative Effect of Top25		4.17	5.488
Percent correctly predicted		0.167	0.167
Observations	1,612	1,612	1,612