Statistical Analysis with R (Estimation of binary logit and probit models)

We use the data on the existence of capital punishment system in U.S. states in “dpdata.txt”. When you open dpdata.txt you see:

```
# Data on the existence of capital punishment system in U.S. states
# D1: Existence of capital punishment,   D1=1: existence, D2=0: non-existence
# T : median length of stay in prison for murderers released from prison
# Y : median of household income
# NW: proportion of non-Caucasian people
D1      T       Y       NW
1       47      1.1     0.321
1       58      0.92    0.224
1       82      1.72    0.127
1       100     2.18    0.063
1       222     1.75    0.021
1       164     2.26    0.027
1       161     2.07    0.139
1       70      1.43    0.218
1       219     1.92    0.008
0       81      1.82    0.012
1       209     2.34    0.076
…………………………….
```

Our aim is to analyze what variable affects the existence of capital punishment in U.S.
After changing the working directory to the one in which the dpdata.txt is saved, you can load the data by typing:

```
  > dpdata=read.table("dpdata.txt",header=T,skip=5)
```

To see the first 8 rows, type

```
 > head(dpdata,8)
```

then you’ll see

```
  D1   T    Y    NW
1  1  47 1.10 0.321
2  1  58 0.92 0.224
3  1  82 1.72 0.127
4  1 100 2.18 0.063
5  1 222 1.75 0.021
6  1 164 2.26 0.027
7  1 161 2.07 0.139
8  1  70 1.43 0.218
```

2. Esimating Logit Model

We let `D1` be dependent variable and other variables be explanatory variables. We estimate the parameters $\beta_0, \beta_1, \beta_2$, and β_3 of

$$\Pr(D1_i=1) = F(\beta_0 + \beta_1 T_i + \beta_2 Y_i + \beta_3 NW_i),$$

where $F(\ )$ is the distribution function of logistic distribution. For this purpose, we use R-function `glm()`. Type

```
> resultlogit=glm(D1~T+Y+NW,family=binomial(link="logit"),data=dpdata)
```

Then, you'll see (it is no problem to ignore the warning message)

```
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
>
```

Note that in the case that you use all variables other than D1 in dpdata as explanatory variables, you can omit explanatory variables by typing "`.`"

```
> resultlogit=glm(D1~.,family=binomial(link="logit"),data=dpdata)
```

Estimation result can be checked by

```
> summary(resultlogit)
Call:
glm(formula = D1 ~ ., family = binomial(link = "logit"), data = dpdata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.52498   0.00000   0.03012   0.34398   1.65742

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.875982   7.427891  -1.599   0.1099
T             0.016051   0.009989   1.607   0.1081
Y             3.941698   3.296000   1.196   0.2317
NW           91.021874  36.347643   2.504   0.0123 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 44.584  on 43  degrees of freedom
Residual deviance: 21.166  on 40  degrees of freedom
AIC: 29.166

Number of Fisher Scoring iterations: 9
```

The values in column of `Estimate` are the estimates of coefficients, those in the column of `Std. Error` are standard errors of those estimates, and `z value` is corresponding to `t value` in regression analyses for examining the null hypothesis that the true value of each coefficient is 0, `Pr( > |z|)` is P-value of `z`. `AIC` is the value called “Akaike Information Criteria” which is a measure of fit of model. Smaller AIC implies better model.

Using these coefficient estimates, we predict the probability of $\mathrm{D1}_i$ being 1 (it is like a fitted value in regression analyses) by

```
 > resultlogit$fitted
```

You’ll see (the values whose actual results are 0 are highlighted in yellow)

```
         1          2          3          4          5          6 
1.00000000 0.99999789 0.99958202 0.98298256 0.62167664 0.89303614 
         7          8          9         10         11         12 
0.99999007 0.99999960 0.48383649 0.09031995 0.99950961 1.00000000 
        13         14         15         16         17         18 
0.86630986 0.80140149 1.00000000 0.71349310 0.99999724 0.17351756 
        19         20         21         22         23         24 
0.12356981 0.98906380 1.00000000 0.41611966 0.99999999 0.52272045 
        25         26         27         28         29         30 
0.95873632 0.99745346 0.94399363 0.99496946 0.99780858 0.99390984 
        31         32         33         34         35         36 
0.98256814 0.25321599 0.98127178 0.42999810 1.00000000 0.93813672 
        37         38         39         40         41         42 
0.99986822 0.99823730 0.35017066 0.99999989 0.86341648 0.21768336 
        43         44 
0.91007074 0.51136739 
```

You can compare these probabilities with actual observations.

```
> dpdata$D1
 [1] 1 1 1 1 1 1
 [7] 1 1 1 0 1 1
[13] 1 1 1 1 1 0
[19] 0 1 1 0 1 1
[25] 0 1 1 1 1 1
[31] 1 1 1 0 1 1
[37] 1 1 0 1 1 0
[43] 1 0
```

You can also check what other results you can see by typing

```
>help(glm)
```

3. Calculating Marginal Probability Effect

Here, we calculate MPE using R. `glm()` function does not automatically calculate MPE, so we have to calculate it by ourselves using the formula. In the case of the logit model, it is relatively easily calculated. Let $\Lambda(x)$ be the distribution function of logistic distribution. Remember that, then, the MPE is

$$MPE_{ij} = f_{\mathrm{logit}}(\beta_0 + \beta_1\mathrm{T} + \beta_2\mathrm{Y} + \beta_3\mathrm{NW})\,\beta_j$$

$$= \Lambda(\beta_0 + \beta_1 \mathrm{T} + \beta_2 \mathrm{Y} + \beta_3 \mathrm{NW}) [1 - \Lambda(\beta_0 + \beta_1 \mathrm{T} + \beta_2 \mathrm{Y} + \beta_3 \mathrm{NW})] \beta_j .$$

where $f_{\text{logit}}(.)$ is the pdf of the logit distribution. Thus, for example, for i = **4** and j = **3**, MPE_{ij} can be calculated by:

```
> Lam=resultlogit$fitted
```

(Remember this is the value of $\Lambda(\beta_0 + \beta_1 \mathrm{T} + \beta_2 \mathrm{Y} + \beta_3 \mathrm{NW})$ in the logit model)

```
> coeflogit=coef(resultlogit)
```

(`coef(result1)` returns a vector of coefficient estimates)

```
> MPElogit=Lam*(1-Lam)*coeflogit[3]
```

("*" is element-by-element product of matrix(or vector))

```
> MPElogit43=MPElogit[4]
```

(`MPE[k]` is k-th element of `MPE`)

The answer is

```
> MPElogit43
       4
 0.06593612
```

The AMPE is the average of MPE, so it can be calculated by

```
> AMPElogit = mean(MPElogit)
> AMPElogit
[1] 0.3045834
```

4. Estimating Probit Model

Similarly, we can estimate probit model too.

```
>resultprobit=glm(D1~.,family=binomial(link="probit"),data=dpdata)
>summary(resultprobit)

Call:
glm(formula = D1 ~ ., family = binomial(link = "probit"), data = dpdata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.33906   0.00000   0.00362   0.34356   1.57845

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.750043   4.041732  -1.670  0.09490 .
T            0.007607   0.005215   1.459  0.14468
Y            2.342983   1.820228   1.287  0.19803
NW          51.802479  19.215316   2.696  0.00702 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 44.584  on 43  degrees of freedom
Residual deviance: 21.402  on 40  degrees of freedom
AIC: 29.402

Number of Fisher Scoring iterations: 10
```

The only difference is that `link="logit"` changes to `link="probit"`.

Again only `NW` is significant at 5% nominal level.

5. Alternative Way to Compute MPE

In Section 3, we used a well known relationship between the distribution function and the probability density function of logistic distribution. In R, there is a function `dlogis(x)`, which returns the value of pdf of the logistic distribution at `x` (if `x` is a vector, then `dlogis(x)` returns a vector of the values of pdf at `x`). In this section, we calculate the MPE with a logit model, using this function. This way of calculation can be easily extended to the probit model.

Recall that MPE of j-th explanatory variable for i-th individual is given by

$$MPE_{ij} = f_{\text{logit}}(\beta_0 + \beta_1 \mathrm{T}_i + \beta_2 \mathrm{Y}_i + \beta_3 \mathrm{NW}_i)\,\beta_j .$$

β_j was already obtained as `coef(j)`. What we have to calculate is the value of $\beta_0 + \beta_1 \mathrm{T}_i + \beta_2 \mathrm{Y}_i + \beta_3 \mathrm{NW}_i$. We calculate this value in the following way (this is a little bit complicated and needs some additional knowledge about R).

Consider a matrix of explanatory variables such as

$$\mathrm{X} = \begin{bmatrix} 1 & \mathrm{T}_1 & \mathrm{Y}_1 & \mathrm{NW}_1 \\ 1 & \mathrm{T}_2 & \mathrm{Y}_2 & \mathrm{NW}_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \mathrm{T}_n & \mathrm{Y}_n & \mathrm{NW}_n \end{bmatrix},$$

where, for example, T_i is a value of T for i-th individual, etc. Then the value of $\beta_0 + \beta_1 \mathrm{T}_i + \beta_2 \mathrm{Y}_i + \beta_3 \mathrm{NW}_i$ is i-th row of Xβ, where β=[β_0, β_1, β_2, β_3]', where A' is the transpose of a matrix(or vector) A. Below, we calculate Xβ.

First, noting that the number of observation (n) is 44, we construct the first column of X (that is n by 1 vector of ones)

```
> const=numeric(44)+1
```

Here `numeric(n)` returns a vector of zeros of length of n. "a +1" adds one to all components of a vector a. Then, matrix X can be constructed by typing

```
> X=cbind(const,dpdata[,2:4])
```

Here `cbind(A, B)` combines two matrices A and B so that `cbind(A, B)`=[A, B], and `A[,a:b]` extracts a- to b-th columns of a data object A.

Next, XB is calculated by typing

```
> XB=as.matrix(X)%*%as.matrix(coeflogit,4,1)
```

Here, `as.matrix(A)` changes a data object A to a matrix object, and `as.matrix(b, c, d)` changes a vector object b to a c by d matrix object so that we can apply matrix calculation to these objects(here R recognizes `as.matrix(X)` as 44 by 4 matrix and `as.matrix(coeflogit,4,1)` as 4 by 1 matrix(vector) so that we can apply the operator "`%*%`", which is for usual matrix product).

Finally, using `dlogis(.)` function, we have

```
> flogis=dlogis(XB)
```

Note that this is equal to `Lam*(1-Lam)` that we calculated in Section 3. Thus, we can calculate MPE_{43}, by

```
> MPElogit2=flogis*coeflogit[3]
> MPElogit2_43=MPElogit2[4]
> MPElogit2_43
[1] 0.06593612
```

We can see that we obtain the same value in Section 3.

Similarly, we can calculate the average MPE by

```
> AMPElogit=mean(MPElogit2)
> AMPElogit
[1] 0.3045834
```

Exercise

R-function "dnorm(x, mean=m, sd=s)" returns the value of the pdf of normal distribution with mean m and standard deviation s at x. Using this function, calculate the MPE for i = 4 and j = 3, for the probit model. Similarly, calculate the average MPE.