Recitation 7 Notes: Binary Dependent Variables in R

Matthew Alampay Davis

November 23, 2021

For these notes, we'll be using the MROZ.dta file included in my recitation folder:

```
mroz <- read.dta13('MROZ.dta')
head(mroz)
```

```
##   inlf hours kidslt6 kidsge6 age educ   wage repwage hushrs husage huseduc
## 1    1  1610       1       0  32   12 3.3540    2.65   2708     34      12
## 2    1  1656       0       2  30   12 1.3889    2.65   2310     30       9
## 3    1  1980       1       3  35   12 4.5455    4.04   3072     40      12
## 4    1   456       0       3  34   12 1.0965    3.25   1920     53      10
## 5    1  1568       1       2  31   14 4.5918    3.60   2000     32      12
## 6    1  2032       0       0  54   12 4.7421    4.70   1040     57      11
##   huswage faminc    mtr motheduc fatheduc unem city exper  nwifeinc      lwage
## 1  4.0288  16310 0.7215       12        7  5.0    0    14 10.910060 1.21015370
## 2  8.4416  21800 0.6615        7        7 11.0    1     5 19.499981 0.32851210
## 3  3.5807  21040 0.6915       12        7  5.0    0    15 12.039910 1.51413774
## 4  3.5417   7300 0.7815        7        7  5.0    0     6  6.799996 0.09212332
## 5 10.0000  27300 0.6215       12       14  9.5    1     7 20.100058 1.52427220
## 6  6.7106  19495 0.6915       14        7  7.5    1    33  9.859054 1.55648005
##   expersq
## 1     196
## 2      25
## 3     225
## 4      36
## 5      49
## 6    1089
```

The relevant variables in the dataset are:

- inlf: "in the labor force" = 1 if women reports working for a wage outside the home at some point during the year, and zero otherwise
- educ: Years of education
- exper: Past years of labor market experience
- expersq: Experience squared
- age: Age in years
- kidslt6: Number of kids younger than six
- kidsge6: Number of kids between six and 18 years old
- nwifeinc: Other sources of income including husband's earnings (in $1000s)

inlf will be our binary outcome variable of interest as we want to estimate the probability of mothers' labor force participation

Data manipulation: Summary statistics by subsamples

Given that the outcome variable is a binary between 0 and 1, we can easily compute the proportion of observations that have a 1 by taking their mean:

```
mean(mroz$inlf)
```

```
## [1] 0.5683931
```

We can also look at counts through the table command:

```
mroz$inlf %>% table
```

```
## .
##   0   1
## 325 428
```

And to convert these to proportions,

```
mroz$inlf %>% table %>% prop.table
```

```
## .
##         0         1
## 0.4316069 0.5683931
```

This gives us the overall proportions, but suppose we also want to get the proportions when splitting by some other variable or condition. Say we want the counts and proportions of mothers' labor force participation split by whether the mother is below or above 45 years old. We can use the filter function to subset:

```
filter(mroz, age >= 45)$inlf %>% table %>% prop.table
```

```
## .
##         0         1
## 0.4631902 0.5368098
```

```
filter(mroz, age < 45)$inlf %>% table %>% prop.table
```

```
## .
##         0         1
## 0.4074941 0.5925059
```

Or equivalently we can do the following using dplyr's group_by argument we learned last week:

```
mroz %>% group_by(age >=45) %>%
  summarize(rate = mean(inlf))
```

```
## # A tibble: 2 x 2
##   `age >= 45`  rate
##   <lgl>       <dbl>
## 1 FALSE       0.593
## 2 TRUE        0.537
```

Implementing linear probability models, probit models, and logit models

Suppose we wanted to regress inlf on nwifeinc, educ, exper, expersq, age, kidslt6, and kidsge6 through these three models.

The linear probability model is straightforward: it's just the same as running standard OLS with/without robust standard errors:

```
mod.lpm <- lm_robust(inlf ~ nwifeinc + educ + exper + expersq + age + kidslt6 + kidsge6,
                     mroz, se_type = 'stata')
summary(mod.lpm)
```

```
##
## Call:
## lm_robust(formula = inlf ~ nwifeinc + educ + exper + expersq +
##     age + kidslt6 + kidsge6, data = mroz, se_type = "stata")
##
## Standard error type:  HC1
##
## Coefficients:
##               Estimate Std. Error t value  Pr(>|t|)   CI Lower   CI Upper  DF
## (Intercept)  0.5855192   0.152260  3.8455 1.306e-04  0.2866098  0.8844287 745
## nwifeinc    -0.0034052   0.001525 -2.2330 2.585e-02 -0.0063988 -0.0004115 745
## educ         0.0379953   0.007266  5.2292 2.214e-07  0.0237310  0.0522596 745
## exper        0.0394924   0.005810  6.7973 2.185e-11  0.0280864  0.0508983 745
## expersq     -0.0005963   0.000190 -3.1384 1.766e-03 -0.0009693 -0.0002233 745
## age         -0.0160908   0.002399 -6.7073 3.922e-11 -0.0208004 -0.0113812 745
## kidslt6     -0.2618105   0.031783 -8.2374 7.892e-16 -0.3242058 -0.1994152 745
## kidsge6      0.0130122   0.013533  0.9615 3.366e-01 -0.0135550  0.0395795 745
##
## Multiple R-squared:  0.2642 ,    Adjusted R-squared:  0.2573
## F-statistic: 62.48 on 7 and 745 DF,  p-value: < 2.2e-16
```

Running the same regression by probit or logit requires a new function called 'glm' that comes in-built in R so no new packages needed. Pay attention to each argument used here:

Probit model:

```
mod.probit <- glm(inlf ~ nwifeinc + educ + exper + expersq + age + kidslt6 + kidsge6,
                     mroz, family = binomial(link = 'probit'))
mod.logit <- glm(inlf ~ nwifeinc + educ + exper + expersq + age + kidslt6 + kidsge6,
                     mroz, family = binomial(link = 'logit'))
```

Before displaying the regression results, notice the additional argument here: family = binomial(link = 'logit'), obviously a bit more difficult to memorize. The reason is that glm is a function that can estimate a wide range of linear models (in fact, the g stands for general). The 'binomial' part is specifying that our outcome variable is binary and the link argument is telling us which binomial model to use. Probit and logit are the most common ones in this category.

Notice we have not specified what type of standard errors to use. We will do this when we display the regression results instead and we refrain from using the summary command to do so, instead using the 'coeftest' command from the *lmtest* package:

```
coeftest(mod.probit, type = 'HC1')
```

```
##
## z test of coefficients:
##
##                Estimate  Std. Error z value  Pr(>|z|)
## (Intercept)  0.27007357  0.50807817  0.5316  0.595031
## nwifeinc    -0.01202364  0.00493917 -2.4343  0.014919 *
## educ         0.13090397  0.02539873  5.1540 2.550e-07 ***
## exper        0.12334717  0.01875869  6.5755 4.850e-11 ***
## expersq     -0.00188707  0.00059993 -3.1455  0.001658 **
## age         -0.05285244  0.00846236 -6.2456 4.222e-10 ***
## kidslt6     -0.86832468  0.11837727 -7.3352 2.213e-13 ***
## kidsge6      0.03600561  0.04403026  0.8177  0.413502
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(mod.logit, type = 'HC1')
```

```
##
## z test of coefficients:
##
##               Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  0.4254524  0.8603645  0.4945  0.620951
## nwifeinc    -0.0213452  0.0084214 -2.5346  0.011256 *
## educ         0.2211704  0.0434393  5.0915 3.553e-07 ***
## exper        0.2058695  0.0320567  6.4220 1.345e-10 ***
## expersq     -0.0031541  0.0010161 -3.1041  0.001909 **
## age         -0.0880244  0.0145729 -6.0403 1.538e-09 ***
## kidslt6     -1.4433541  0.2035828 -7.0898 1.343e-12 ***
## kidsge6      0.0601122  0.0747893  0.8038  0.421539
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

'HC1' refers to the heteroskedasticity-robust standard errors we've been using all semester, namely the ones that come from Stata. In fact, you can use "se_type = 'HC1' " as an argument in lm_robust in place of "se_type = 'stata' " and you'll get the same thing.

Creating predictions

We now have three regression models with which we can generate predicted values for any parametrization of the regressors. Suppose we're interested in the predicted labor force participation of a hypothetical mother with median values for all the regressors, which is to say we have an imaginary mother who has the median level of education, experience, age, number of kids younger than six, number of kids between six and 18, and other sources of income. We can generate such an observation by running the following:

```
median.mother <- summarize_all(mroz, .funs = c('median'))
median.mother
```

```
##   inlf hours kidslt6 kidsge6 age educ wage repwage hushrs husage huseduc
```

```
## 1    1   288       0       1  43   12   NA       0   2164     46      12
##   huswage faminc    mtr motheduc fatheduc unem city exper nwifeinc lwage
## 1  6.9758  20880 0.6915       10        7  7.5    1     9     17.7    NA
##   expersq
## 1      81
```

We can use all three regression models to create a prediction for this hypothetical person's probability of participating in the labor force using the 'predict' command:

```
predict(mod.lpm, newdata = median.mother)
```

```
##         1
## 0.6094292
```

```
predict(mod.logit, newdata = median.mother, type = 'response')
```

```
##         1
## 0.6397075
```

```
predict(mod.probit, newdata = median.mother, type = 'response')
```

```
##         1
## 0.6363523
```

Encouragingly, these predictions are pretty close to one another, the logit and probit models especially as will usually be the case. Note that when we create predictions using the logit and probit models, we want to specify type = 'response'. This is because the coefficients that come out of these non-linear regressions must be transformed before they can be directly interpreted in terms of the units of the response/outcome variable. See the lecture notes and textbook for more on interpreting coefficients in a logit or probit model, it is a very plausible exam-type question.

Suppose we have new data on a new mother named Emily with the following profile:

- educ: 16
- exper: 4
- expersq: 4^2
- age: 26
- kidslt6: 0
- kidsge6: 0
- nwifeinc: 0

And now suppose we were interested in the change in the predicted probability of participating labor force if Emily had a two-year-old child. We could input the two cases in the following way:

```
cases <- data.frame(educ = c(16,16),
                    exper = c(4,4),
                    expersq = c(4^2,4^2),
                    age = c(26, 26),
                    kidslt6 = c(0,1), # the only difference
                    kidsge6 = c(0,0),
                    nwifeinc = c(0,0))
cases
```

```
##   educ exper expersq age kidslt6 kidsge6 nwifeinc
## 1   16     4      16  26       0       0        0
## 2   16     4      16  26       1       0        0
```

Then we could compute predictions for the two versions of Emily for each model

```
# LPM
pred.lpm <- predict(mod.lpm, cases)
pred.lpm
```

```
##         1         2
## 0.9235117 0.6617012
```

```
# Probit
pred.probit <- predict(mod.probit, cases, type = 'response')
pred.probit
```

```
##         1         2
## 0.9269671 0.7208083
```

```
# Logit
pred.logit <- predict(mod.logit, cases, type = 'response')
pred.logit
```

```
##         1         2
## 0.9204559 0.7320803
```

Since there are only two cases we're comparing, we can easily take their differences:

```
# LPM
diff(pred.lpm)
```

```
##          2
## -0.2618105
```

```
# Probit
diff(pred.probit)
```

```
##          2
## -0.2061588
```

```
# Logit
diff(pred.logit)
```

```
##          2
## -0.1883756
```

The findings here suggest that someone with Emily's profile having one child younger than age 6 decreases the predicted probability of participating in the labor force by between 18.9-26.2 percentage points from a base probability of about 92-93%.

Statistics for logit/probit models

The Pseudo-R^2

To get pseudo-R^2's for the probit or logit models, we'll need the aptly named 'pR2' function from the *pscl* package

```
require(pscl)
pR2(mod.logit)
```

```
## fitting null model for pseudo-r2
```

```
##          llh      llhNull           G2     McFadden          r2ML          r2CU
## -401.7651511 -514.8732046  226.2161069    0.2196814    0.2594927    0.3481893
```

The statistic we want is the McFadden one.

Alternatively, we can use the following equivalent formula:

```
1-mod.logit$deviance/mod.logit$null.deviance
```

```
## [1] 0.2196814
```

Confidence intervals

For the LPM, this is straightforward:

```
confint(mod.probit, level = 0.99)
```

```
## Waiting for profiling to be done...
```

```
##                    0.5 %        99.5 %
## (Intercept) -1.039507869  1.5820770689
## nwifeinc    -0.024629861  0.0003244882
## educ         0.066625949  0.1968067625
## exper        0.075513565  0.1720437944
## expersq     -0.003437826 -0.0003386041
## age         -0.074968786 -0.0312693206
## kidslt6     -1.179339191 -0.5686966380
## kidsge6     -0.075842055  0.1482326997
```

For the logit/probit models, marginal effects will vary depending on the values of the regressors since they are non-linear models. We will require the 'sandwich' package to select the type of standard errors we want ('HC1') and the 'margins' package to evaluate emarginal effects at particular values. Here, we'll compute marginal effects at median values since we've already created the profile of the hypothetical median mother above.

```r
# Standard errors
require(sandwich)
logit.se <- vcovHC(mod.logit, type = 'HC1')
probit.se <- vcovHC(mod.probit, type = 'HC1')

# Compute marginal effects at median values
require(margins)
margins(mod.logit,
        data = median.mother,
        vcov = logit.se) %>%
  summary
```

```
##    factor     AME     SE       z      p   lower   upper
##       age -0.0203 0.0033 -6.0679 0.0000 -0.0268 -0.0137
##      educ  0.0510 0.0105  4.8424 0.0000  0.0303  0.0716
##     exper  0.0474 0.0072  6.6355 0.0000  0.0334  0.0615
##   expersq -0.0007 0.0002 -3.2238 0.0013 -0.0012 -0.0003
##   kidsge6  0.0139 0.0186  0.7440 0.4569 -0.0226  0.0504
##   kidslt6 -0.3327 0.0440 -7.5617 0.0000 -0.4189 -0.2464
##  nwifeinc -0.0049 0.0021 -2.3718 0.0177 -0.0090 -0.0009
```

The 'lower' and 'upper' columns here match Stata's confidnece intervals (i.e., at the 95% level)

Evaluating the models

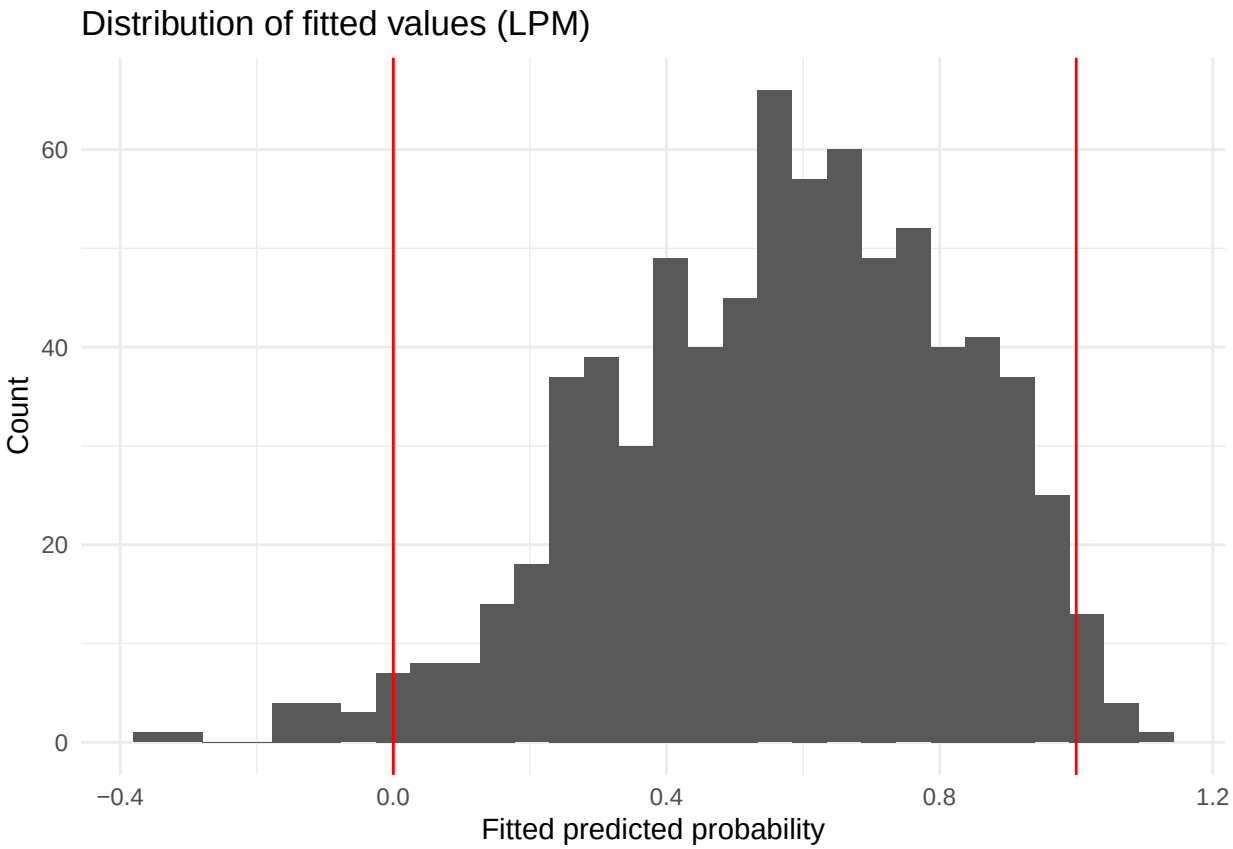
For each observation in our dataset, we can use the estimated regressions to produce fitted values or predictions given the profile given. We can call these directly from the model object.

For the linear probability model, we might be interested in seeing how many fitted values are contained within the probability bounds [0,1]. Let's plot these:

```r
fit.lpm <- data.frame(y.hat = mod.lpm$fitted.values)
fit.probit <- data.frame(y.hat = mod.probit$fitted.values)
fit.logit <- data.frame(y.hat = mod.logit$fitted.values)

ggplot(fit.lpm, aes(x = y.hat)) +
  geom_histogram() + # Plot a histogram
  theme_minimal() + # Optional aesthetic theme
  ggtitle('Distribution of fitted values (LPM)') +
  ylab('Count') +
  xlab('Fitted predicted probability') +
  geom_vline(xintercept = 0, col = 'red') +
  geom_vline(xintercept = 1, col = 'red')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As we can see, there are quite a few predicted probabilities that don't make real sense in the sense that they aren't within [0,1]. This is a limitation of the LPM and in general we're OK with using LPM as long as not too many observations lie outside the bounds of the red lines. We can see how many do:

```
table(mod.lpm$fitted.values < 0 | mod.lpm$fitted.values > 1)
```

```
##
## FALSE  TRUE
##   720    33
```

33 out of 753 observations have a nonsensical predicted probability (the | bar means OR). In percentages:

```
table(mod.lpm$fitted.values < 0 | mod.lpm$fitted.values > 1) %>%
  prop.table
```

```
##
##     FALSE      TRUE
## 0.9561753 0.0438247
```

4.4% of the fitted values are nonsensical. An exam question might ask you whether this is enough to dismiss the linear probability model. Obviously, the probit and logit models by construction will have zero such observations:

```
table(mod.probit$fitted.values < 0 | mod.probit$fitted.values > 1)
```

```
##
## FALSE
##   753
```

```
table(mod.logit$fitted.values < 0 | mod.logit$fitted.values > 1)
```

```
##
## FALSE
##   753
```

We can also evaluate these models by how many predictions are 'correct' in the sense of whether all predicted probabilities > 0.5 correspond to a true value of 1 and all predicted probabilities < 0.5 correspond to a true value of 0. Many ways of doing this, here's one using the fact that numbers below 0.5 will round to 0 while those above will round to 1:

LPM:

```
table(round(mod.lpm$fitted.values), mroz$inlf) %>%
  prop.table
```

```
##
##             0         1
##   0 0.2695883 0.1035857
##   1 0.1620186 0.4648074
```

The 0,0 and 1,1 cells are correct predictions, the other cells are incorrect. We can sum the diagonals to get the proportion correctly predicted:

```
table(round(mod.lpm$fitted.values), mroz$inlf) %>%
  prop.table %>%
  diag %>%
  sum
```

```
## [1] 0.7343958
```

Probit:

```
table(round(mod.probit$fitted.values), mroz$inlf) %>%
  prop.table %>%
  diag %>%
  sum
```

```
## [1] 0.7343958
```

Logit:

```
table(round(mod.logit$fitted.values), mroz$inlf) %>%
  prop.table %>%
  diag %>%
  sum
```

```
## [1] 0.7357238
```

By this measure, the LPM and probit models correctly classified the same number of observations: 73.4%. The logit model did only a tiny bit better: 73.6%.