**Lab Assignment Week 11**

**I. Objectives**: The objective is to compare three models for binary (or dichotomous) dependent variables: the linear probability model, logit, and probit.

**II. Datasets**: *conf06.dta*

**III. Packages**: *foreign*, *car*, *stargazer, whitetest.R*

**IV. Preparation**

1) Download the paper that uses this data – “Epstein et al 2006 JOP.pdf” – and review the paper, especially examining the tables and finding the variables’ names

2) Find a ‘cumulative standard Normal probability table’ that translates *z* scores into percentiles

3) Open RStudio by double-clicking the icon or selecting RStudio from the Windows Start menu.

3) Clear any data in memory by typing rm(list=ls())

4) Download dataset *conf06.dta* and place it in a working directory

5) Download R scripts “Lab11.R” and “white-test.R” and place them in a working directory.

6) Open the R script by typing *Ctrl*+*O* or by clicking on File in the upper-left corner, using the dropdown menu, and navigating to the script in your working directory.

7) Run lines 3–5 in the R script to load four packages that you will need. If any of these are not already installed, you need to install them using the **Packages** tab. By removing the # from the beginning of line 3 you will install *stargazer*, which I believe is the only package we have not used yet.

**V. Instructions for Lab Week 11**

You will use data on senators’ votes on Supreme Court nominees, which we might use in Lecture 23. The key explanatory variables are the squared distance between the senator’s ideal point and the nominee’s inferred ideal point (EuclDist2), the nominee’s perceived qualifications (qual) or lack of qualifications (lackqual), a dummy variable indicating that the president is strong (strngprs), and a dummy variable indicating that the senator shares the president’s party affiliation (sameprty).

Load the dataset by typing the following code or running lines 7−8, changing the directory if needed:

> data <- read.dta("C:/conf06.dta")

> conf06 <- subset(data, data$nominee!="ALITO")

The second line of code drops the observations for Justice Alito, because the published paper used only nominees through Chief Justice John Roberts. Lines 9–10 make the data frame that you will use, named conf, thinner by selecting only a handful of variables. Lines 11–12 are included to convert two variables that were factors (vote and strngprs) into numeric variables.

To check that your cases are identical to those used in the paper, run line 13 or line 14 to replicate Table 1 (p. 300). (The code in line 13 ought to work, but if it does not, the extra as.data.frame() command may be necessary.)

Run lines 17−20 to perform a simple analysis of how likely senators are to vote in favor of a Supreme Court nominee depending on presidential strength (separate tables) and senators’ party affiliations (the separate columns in each table).[[1]](#footnote-1) These proportions do not control for the nominee’s qualifications or for the difference between a nominee’s ideological position and each individual senator. Nevertheless, it is worthwhile to fill in the proportions of yes votes (the **bottom row** in each of the tables) in the following four conditions:

Proportion of yes votes

1. Weak president and an opposing-party senator
2. Weak president and a senator from the president’s party
3. Strong president and an opposing-party senator
4. Strong president and a senator from the president’s party

You might notice that the effects are not constant: differences in party affiliation have greater effects when the president is weak (+\_\_\_\_ %) compared to when the president is strong (+\_\_\_\_ %).

Instead, you might switch the positions of strngprs and sameprty to create separate tables depending on whether senators belong to the same party as the president or the opposing party, and include weak and strong presidents as separate columns in the same table. Use the code shown in lines 22–25 to create the tables I just described.

You might notice instead that presidential strength has less impact on the votes of same-party senators (+\_\_\_\_ %) than opposing-party senators (+\_\_\_\_ %). It may be worth spending a few minutes pondering whether these effects necessitate including an interaction term in the model and/or whether they result purely from so-called “compression.”

Run lines 28−29 to replicate the analysis in column 2 of Table 2, on page 300 of the paper, using probit.

It is worth mentioning that we are using a variable called lackqual, which equals 1 minus qualifications (which are scaled from 0 to 1). The reason for doing this is ease of comparison. In theory, a senator should be less likely to vote for a nominee who is further away in ideological space (a higher value of EuclDist2) or who is less qualified (a higher value of lackqual), suggesting negative coefficients for each variable.

Next, run lines 31–36 to find and name the mean, minimum, and maximum values to two variables – lackqual and EuclDist2. You should observe that the mean value of lackqual is .22 and the mean value of EuclDist2 is .18.

Lines 38−43 show how to generate predicted probabilities of a yes vote at varying levels of qualifications. The following computations assume the nominee and senator are separated by the average ideological distance (EuclDist2 = .1823883), that the president is weak (strngprs = 0), and that the senator belongs to the different party from the president (sameprty = 0). Estimate predicted probability of a yes vote in the following three conditions:

Linear value Predicted probability

1. The most qualified nominee (lackqual = 0): (39)
2. An average qualified nominee (lackqual ≈ .22): (41)
3. The least qualified nominee (lackqual = .89): (43)

The commands for linear predictions are lines 38, 40, and 42. Each time you compute the linear combination, you need to find the predicted probability either by reference to a standard cumulative normal table or by entering that number into the following: pnorm(), as appears in 39, 41, and 43.

Lines 45−50 provide an alternative way of calculating predicted probabilities using the predict command. Advantages of this latter approach are that it also produces a standard error, which you can use to produce a confidence interval, and that these commands can efficiently create a table of predicted values (and standard errors) ifyou create a new data frame out of pre-determined values for the explanatory variables.

Now turn your attention to the predicted effects of changing the two qualitative independent variables — presidential strength (strngprs) and party affiliation (sameprty).

For calculating the next set of predicted probabilities, set the lack of qualifications at its mean value (≈ .22) and ideological distance at its mean value (≈ .18). Lines 52−59 generate the entries that you would need to fill in a table with predicted probabilities of a yes vote in the following four conditions:

Predicted probability

1. Weak president and an opposing-party senator \_\_\_\_
2. Weak president and a senator from the president’s party \_\_\_\_
3. Strong president and an opposing-party senator) \_\_\_\_
4. Strong president and a senator from the president’s party \_\_\_\_

Use the predicted probabilities above to fill in the following table of marginal effects:

|  |  |
| --- | --- |
| To assess the marginal impact of changing… | Compare… |
| … party affiliation when the president is weak | b) – a) = |
| … party affiliation when the president is strong | d) – c) = |
| … presidential strength for senators in the opposing party | c) – a) = |
| … presidential strength for senators in the president’s party | d) – b) = |

Run lines 62−63 to repeat the analysis with a logit (aka logistic regression) model. Later we will conduct a more thorough comparison of logit and probit coefficients, standard errors, and z-statistics (note that *z* replaces *t*, since we are now using the Normal distribution rather than Student’s *t* distribution).

Run lines 65−72 to use the probit estimates to calculate the following:

linear values & predicted probabilities:

1. Weak president and an opposing-party senator \_\_\_
2. Weak president and a senator from the president’s party \_\_\_
3. Strong president and an opposing-party senator) \_\_\_
4. Strong president and a senator from the president’s party \_\_\_

Use the predicted probabilities above to fill in the following table of marginal effects:

|  |  |
| --- | --- |
| To assess the marginal impact of changing… | Compare… |
| … party affiliation when the president is weak | f) – e) = |
| … party affiliation when the president is strong | h) – g) = |
| … presidential strength for senators in the opposing party | g) – e) = |
| … presidential strength for senators in the president’s party | h) – f) = |

Run line 75–76 to repeat the analysis with a linear probability model. Note that unlike the logit and probit models, the linear probability model generates marginal effects that are *unconditional*. In other words, the marginal effect of changing party affiliation will be the same whether the president is weak or strong, and will equal the regression coefficient (βsameprty = \_\_\_\_\_). Likewise, the marginal effect of changing presidential strength will be the same regardless of which party the senator belongs to, and will equal the regression coefficient (βstrngprs = \_\_\_\_\_).

Each time you compute the linear combination, this is your predicted probability. So, when you run lines 78−81, no additional steps are required. Just for purposes of comparison, let’s fill in the table again for the same four conditions as before:

Predicted probability

1. Weak president and an opposing-party senator \_\_\_\_
2. Weak president and a senator from the president’s party \_\_\_\_
3. Strong president and an opposing-party senator) \_\_\_\_
4. Strong president and a senator from the president’s party \_\_\_\_

As we have discussed in lecture, the linear probability model has a few advantages (unconditional marginal effects and easy-to-compute predicted probabilities). However, the linear probability model also has some potential disadvantages that we need to investigate.

First, does the model yield any out-of-bounds predictions? Run line 83 to see the range of predicted values, and then run line 84 to see how many observations exceed 1? Roughly how many?

Second, does the model yield homoscedastic errors? Find and run the R script, **white-test.R**, and then run line 85 to conduct White’s test (using the R script that I uploaded earlier in the semester); do you reject the null hypothesis of homoscedasticity? You can run lines 86–87 to examine the residuals versus the values of the EuclDist2 and lackqual variables.

Given that the potential problems with a linear probability model are realized in this example, discuss in lab what you could or should do – other than using logit and probit – to address these problems.

Finally, to examine how well each model does of predicting the outcome, we’ll focus on the “correct classification” statistic.

For the regression, you’d have to create the predictions somewhat laboriously by generating the predicted values, and then generating a new binary variable that equals zero if the predicted value is less than 0.5 and equals one if the predicted value is greater than or equal to 0.5 (lines 90−91). Create a table using the predicted probabilities and the actual value of vote by running line 92.

For the logit model, you can run line 94 to create predicted probabilities and then code them as a binary variable that equals zero if and only if the predicted value is less than 0.5; line 95 creates a table with correct predictions on the main diagonal and incorrect predictions off the main diagonal.

Lines 97−98 duplicate this process for the probit model.

If you are interested in the improvement that your model contributes, you can estimate the null logit model in line 100 (and the null probit model in line 104); these commands run models with only the value of 1 as the independent variable, and no other explanatory variables. Lines 101−102 create predicted values, code them as binary, and then identify incorrect and correct predictions for the logit model (lines 105−106 duplicate this process for the probit model).

Our models increase the percent correctly predicted from 88% to about 91½%. A net improvement of 3½% may not sound like much, but the *proportional* reduction in error is nearly 29%: (.9148−.8803) ÷ (1−.8803) = .288

In line 108, we use the stargazer package compare the estimates from the linear probability model, probit, and logit, respectively. It will display the coefficients and standard errors in a single table. This is one convenient way to examine the statistical significance of different variables, and in particular whether the significance changes across estimators.

Some textbooks list rough “conversion factors” for comparing coefficient sizes across estimators. For example, logit coefficients are typically 4 times as large as LPM coefficients, and are about 1.6 times as large as probit coefficients; probit coefficients are typically 2.5 times as large as LPM coefficients. How well do those relationships hold up with these data?

To clear the **Environment**, type rm(list=ls()), as shown in line 110, or click on the broom icon.

To clear the **Console** window, type Ctrl-*l*

1. That is, this analysis does not take into account the characteristics of the individual nominee. A form of analysis that uses aggregated data like this is called “grouped logistic regression.” [↑](#footnote-ref-1)