PPOL561 | Summary
Multinomial Dependent Variables

Overview

The following offers a summary of main points when dealing with multinomial dependent variables. These notes summarize some of the main points from the Long reading and the lectures.

Multinomial Responses

Nomial outcome variable is a variable with discrete categories but no intrinsic ordering. We mainly see this when a variable is just "names" or stated preferences. The focus is analyzing questions of "what", "which", "who", and "where", rather than questions regarding "how much" or "how many".

Examples:

- Who did you vote for?
- Which product did you purchase?
- What kind of job do you have?
- What is the ethnicity of a new hire?

Why not OLS

OLS makes no sense here given there is not continuous interval that can be assumed. Doing so would require that we impose some arbitrary ordering on the categories (bananas are better than apples, which are better than pears). Results could differ drastically if one were to assume a different ordering.

Alternatively, one could consider running individual logits for each category (see below). If so, then the same problems discussed with respect to using OLS for binary response variables will be at play.

Multinomial Dependent Variable Model

The multinomial logit model (MNLM) can be thought of as simultaneously estimating binary logits for all possible comparisons among the outcome categories. In this sense, multinomial logit is a simple extension of the binary logit. That's nice!

However, running a bunch of logit models and compare the results to all possible alternative outcomes would becomes quickly intractable (and redundant). Rather we can estimate the choice of any one category all in one go.

For example, consider three candidates in an election: Bruce (B), Jack (J), and Sally (S). We want to build a model that calculates the probability that a voter wil choose one of them. What is the probability of voting for Sally given one can also vote for Bruce or Jack?

Consider treating this as a binary response, rather a categorical one. That is, rather than calculating the probability of voting for Bruce, Jack, or Sally, we just calculated the probability of voting for Sally (1) or not (0). If this were the case, we could use a logit or probit model.

$$pr(vote_{sally} = 1) = \frac{exp(x_i\beta^{sally})}{1 + exp(x_i\beta^{sally})}$$

Likewise we could do the same for Bruce.

$$pr(vote_{bruce} = 1) = \frac{exp(x_i\beta^{bruce})}{1 + exp(x_i\beta^{bruce})}$$

And so on.

The binary response model allows us to model the probability as a linear combination of covariates. As we saw, the logit function converts those covariates into a probability space that we can the estimate coefficients for using Maximum Likelihood. One thing to note here is that we'll get a different set of coefficients for each logit model, i.e. β^{sally}, β^{bruce}, β^{jack}, ect.

For multinomial models, we are largely doing the same thing. The aim, again, is to model the decision between choices using a linear combination of covariates and then transforming those covariates using a link function. However, in the multinomial context we can estimate the probability of making one choice given all the alternative options. To do this, we'll use a more complex form of the logit model where we'll model all three choices simultaneously.

$$pr(vote = sally) = \frac{exp(x_i\beta^{sally})}{exp(x_i\beta^{bruce}) + exp(x_i\beta^{jack}) + exp(x_i\beta^{sally})}$$

Right now this model isn't identified (meaning there could be many solutions to it, see Long Ch. 6). To identify the model, we need to constrain the coefficients for one of the choices (i.e. candidates) to equal 0. This choice will be the **reference category**.

From our candidates, let's choose Bruce to be the reference category: $\beta^{bruce} = 0$. Thus, the multinomial logit model is as follows:

$$pr(vote = sally) = \frac{exp(x_i\beta^{sally})}{1 + exp(x_i\beta^{jack}) + exp(x_i\beta^{sally})}$$

If we wanted to examine the probability of voting for any of the other candidates, we'd just re-arrange.

$$pr(vote = jack) = \frac{exp(x_i\beta^{jack})}{1 + exp(x_i\beta^{jack}) + exp(x_i\beta^{sally})}$$

$$pr(vote = bruce) = \frac{1}{1 + exp(x_i\beta^{jack}) + exp(x_i\beta^{sally})}$$

Assumption

A key assumption for a multinomial choice model is the **independence of irrelevant alternatives assumption (IIA)**. The idea is that the relative probability of existing alternatives (choices) is not affected by changes to the choice set (i.e. adding or removing choices). Essentially, we're assuming that the errors across choices are not correlated, so adding and dropping a choice doesn't impact the underlying probability of the choices left on the table. If this assumption is violated, estimates will be biased.

For example, say we had commuters deciding between three modes of travel: bus, train, and car. The probability are car (.5), train (.33), and bus (.17). Thus, the odds of choosing the bus over the train are $\frac{.17}{.33} = .52$. The odds of choosing the bus over the car $\frac{.17}{.5} = .34$.

Assuming the color of the bus (red or blue) doesn't change the probability that you'll select a train or car, then we can add the choice of a red bus and blue bus to the menu, which would effectively split the probability of the bus in half between the two: $pr(bus_{blue}) = .085$ and $pr(bus_{red}) = .085$.

IIA can be a hard assumption to make because we could think of scenarios where this assumption appears to be violated. Again, let's turn to our candidates from above. Say $pr(vote = sally) = .6$ and the $pr(vote = Bruce) = .4$; however, then late in the game Jack enters the race. Since he is closer to Sally on the political spectrum, he draws from her support. The probability of choosing Jack is still low $pr(vote = Jack) = .05$; however, it alters the underlying probability of voting for Sally ($pr(vote = sally) = .55$) and thus violates the IIA.

There are tests to check for this, but none 100% perfect (See the Long reading). The IIA assumption is something we need to think and theorize about. Like the parallel regression assumption from ordered probit, we need to use theory to guide us.[1]

[1]For example, let's consider the problem a little differently. Let's say there is always a lurking fourth choice: not to vote at all. The probabilities are then: $pr(vote = sally) = .5$, $pr(vote = Bruce) = .3$, and $pr(vote = abstention) = .2$. In this configuration, the entry of a new candidate could split the abstention bin. That is, the entry of a new candidate doesn't change the probability that someone will vote for either of the two other candidates, but that it'll reduce the probability of abstention. The set up would be $pr(vote = sally) = .5$, $pr(vote = Bruce) = .3$, $pr(vote = jack) = .05$, and $pr(vote = abstention) = .15$. If what we care about is estimates on the probability that one votes for a specific candidate, then this split doesn't change anything with regard to the estimates of interest. This example isn't perfect, but it's a way of using theory to think through things and justify decisions.

Computation

Multinomial models can be computationally-demanding to compute. Why? Recall that we are estimating coefficients for each and every choice. So in the candidate example we have two sets of estimates for β^{jack} and β^{sally}. This can demand a lot of resources to optimize as the number of choices increase and/or the number of covariates we input increases. Keep this in mind if you ever try to run a complex model and it takes a while to run!

Also, note the emphasis on the logit model here and not the probit. A multinomial probit model is far more complex (these models allow the errors to be correlated), which requires more computation. Though there are packages that estimate this model, we'll focus on the multinomial logit as it runs quicker and is easier to conceptualize.

Connection to Ordered Outcomes

For the ordered probit/logit, we needed to make the parallel regressions assumption. That assumption stated that coefficients remained the same across the cutpoint categories (i.e. the effect of x on y is constant as we move from one cut point to the next). Here we are throwing that assumption out the window and estimating a different set of coefficients for each choice. Thus, one may use a multinomial choice model for ordered data if he/she believes the parallel regressions assumption is suspect, or the underlying ordering is ambiguous. As we'll see below, get very similar results from both models in terms of predictions.

Calculating Predicted Probabilities

Calculating predicted effects is a little more involved in the multinomial setting. In our candidate example from above, say x_i is a dummy variable that takes on the value of 0 and 1. To calculate the probability of voting for Sally given $x_i = 1$ we do as follow:

$$pr(vote = sally) = \frac{exp(\beta_0^{sally} + \beta_1^{sally} x_i)}{1 + exp(\beta_0^{jack} + \beta_1^{jack} x_i) + exp(\beta_0^{sally} + \beta_1^{sally} x_i)}$$

$$pr(vote = sally | x = 1) = \frac{exp(\beta_0^{sally} + \beta_1^{sally}(1))}{1 + exp(\beta_0^{jack} + \beta_1^{jack}(1)) + exp(\beta_0^{sally} + \beta_1^{sally}(1))}$$

$$pr(vote = sally | x = 0) = \frac{exp(\beta_0^{sally} + \beta_1^{sally}(0))}{1 + exp(\beta_0^{jack} + \beta_1^{jack}(0)) + exp(\beta_0^{sally} + \beta_1^{sally}(0))}$$

As before, we'll hold all other covariates at their observed values.

If we want to know how the probability of voting for Bruce changes as x shifts from 0 to 1, we just re-arrange the equation as we saw above.

$$pr(vote = Bruce|x = 1) = \frac{1}{1 + exp(\beta_0^{jack} + \beta_1^{jack}(1)) + exp(\beta_0^{sally} + \beta_1^{sally}(1))}$$

$$pr(vote = Bruce|x = 0) = \frac{1}{1 + exp(\beta_0^{jack} + \beta_1^{jack}(0)) + exp(\beta_0^{sally} + \beta_1^{sally}(0))}$$

And so on. The point being is that the process is more involved because we have to calculate the probability for every choice given every manipulation, but the logic that we have for doing this with binary and ordered models still holds.

Likewise, if we wanted to calculate the discrete difference for voting for Sally given x, we can simply calculate the difference in probabilities as we did with other models.

$$pr(vote = sally|x = 1) - pr(vote = sally|x = 0)$$

Example: Student Support for the Iraq War

Consider the following survey data draw from a sample of 500 college students in 2002 leading up to the Iraq war. The survey asked whether students agreed with the United States entering into the Iraq War. The data contains three variables:

```
dat <- read_csv('student_vote.csv')
dat$warsup = factor(dat$warsup,levels=c('strongly oppose',
                                        "somewhat oppose",
                                        "somewhat support",
                                        "strongly support"))
summary(dat)
```

```
##               warsup         dem            female
##  strongly oppose :168   Min.   :0.000   Min.   :0.000
##  somewhat oppose : 98   1st Qu.:0.000   1st Qu.:0.000
##  somewhat support:126   Median :1.000   Median :1.000
##  strongly support:108   Mean   :0.536   Mean   :0.576
##                         3rd Qu.:1.000   3rd Qu.:1.000
##                         Max.   :1.000   Max.   :1.000
```

- `warsup`: whether the respondent somewhat or strongly oppose/supported the War in Iraq
- `dem`: 1 if the repondent was a democrat, 0 otherwise.
- `female`: 1 if the repondent is female, 0 otherwise.

The dependent variable is `warsup`, and the key independent variable that we're going to look at is `dem`. Our hypothesis is that democratic respondents are less likely to support the Iraq War than non-democratic respondents.

We already encountered these data in the Ordered write-up. This time, let's treat the model as a multinomial choice model rather than an ordered probit (why would we do this?).

Note that since `strongly oppose` is the first factor it'll be our reference category.

```
levels(dat$warsup)
```

```
## [1] "strongly oppose"  "somewhat oppose"  "somewhat support"
## [4] "strongly support"
```

We can estimate multinomial logits using the `multinom()` from the `nnet` package.[2]

```
multi_mod <- nnet::multinom(warsup ~ dem + female,
                            data=dat)
```

```
## # weights:  16 (9 variable)
## initial  value 693.147181
## iter  10 value 578.220271
## final  value 571.563301
## converged
```

```
multi_mod
```

```
## Call:
## nnet::multinom(formula = warsup ~ dem + female, data = dat)
##
## Coefficients:
##                   (Intercept)        dem       female
## somewhat oppose    0.04525367 -0.4162225 -0.407679600
## somewhat support   1.20336368 -2.5008800  0.007914799
## strongly support   1.59309190 -3.8175076 -0.550507756
##
## Residual Deviance: 1143.127
## AIC: 1161.127
```

Like with the binary and ordered models, the coeffients report the log odds, but cannot be directly interpreted. Moreover, recall that these coefficient estimates are relative to the baseline category, which complicates interpretation even further. We can note the general direction of the coefficients. The coefficients for `dem` become increasingly more negative as we move across choices.

One interesting thing to note here is the change in the magnitude of the coefficients as one moves from oppose to strongly support. One question this begs is whether the parallel regression assumption, which we made in the context of the ordered model, was appropriate in this case.

[2]Note that the `nnet` package is actually estimating the model using a Neural Network, which is done to help estimate complicated multi-class problems. There is another package `mlogit` that estimates the models similiar to our conversation above; however, mlogit requires that the data be reformatted a particular way. So to minimize confusion, we'll use the `nnet` package instead.

```
# Let's use broom to get the standard errrors
broom::tidy(multi_mod,
            exponentiate = F # Don't auto exp!
            ) %>%
  mutate_if(is.numeric, function(x) round(x,3))
```

```
## # A tibble: 9 x 6
##   y.level          term        estimate std.error statistic p.value
##   <chr>            <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 somewhat oppose  (Intercept)    0.045     0.32      0.141   0.888
## 2 somewhat oppose  dem           -0.416     0.325    -1.28    0.2
## 3 somewhat oppose  female        -0.408     0.261    -1.56    0.119
## 4 somewhat support (Intercept)    1.20      0.274     4.40    0
## 5 somewhat support dem           -2.50      0.289    -8.66    0
## 6 somewhat support female         0.008     0.276     0.029   0.977
## 7 strongly support (Intercept)    1.59      0.273     5.83    0
## 8 strongly support dem           -3.82      0.384    -9.94    0
## 9 strongly support female        -0.551     0.306    -1.80    0.072
```

It appears that the coefficient on `dem` for the "somewhat oppose" choice is not statistically significant at the .05 level. The coefficients for `dem` on the other choices, however, are statisticaly signficiant at conventional levels.

Predicted Probabilities

As we saw above, calculating the predicted probabilities for any one choice in a multinomial logit is a little more involved, but follows the same basic logic from before.

Extract the coefficients

```
B_all <- coefficients(multi_mod)
B_so = B_all[1,] # Coefs: somewhat oppose
B_ss = B_all[2,] # Coefs: somewhat support
B_SS = B_all[3,] # Coefs: strongly support
```

Extract the model matrix

```
X = model.matrix(multi_mod)
head(X)
```

```
##   (Intercept) dem female
## 1           1   0      0
## 2           1   0      1
## 3           1   0      1
## 4           1   1      1
## 5           1   0      0
```

```
## 6           1    0       0
```

Manipulate the value of the key independent variable, which here is being a democrat or not.

```
X[,2] = 0 # Not a democrat
```

Calculate the predicted probabilities for each choice.

```
pr_strg_opp = 1/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_some_opp = exp(X%*%B_so)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_some_supp = exp(X%*%B_ss)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_strg_supp = exp(X%*%B_SS)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
```

Now for the alternative: respondent self-identifies as a democrat.

```
X[,2] = 1 # Democrat
pr_strg_opp_dem = 1/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_some_opp_dem = exp(X%*%B_so)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_some_supp_dem = exp(X%*%B_ss)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
pr_strg_supp_dem = exp(X%*%B_SS)/(1+exp(X%*%B_so) + exp(X%*%B_ss) + exp(X%*%B_SS))
```

Bring together as a table.

```
categories = c("Strongly Oppose","Somewhat Oppose",
               "Somewhat Support","Strongly Support")
tab1 <-
  tibble(categories,
         democrat = 0,
         props = c(mean(pr_strg_opp),
                   mean(pr_some_opp),
                   mean(pr_some_supp),
                   mean(pr_strg_supp)))

tab2 <-
  tibble(categories,
         democrat = 1,
         props = c(mean(pr_strg_opp_dem),
                   mean(pr_some_opp_dem),
                   mean(pr_some_supp_dem),
                   mean(pr_strg_supp_dem)))

# Bring together
pred_effects = bind_rows(tab1,tab2)

# Print off
pred_effects %>%
  arrange(categories) %>%
  mutate(props = paste0(round(props,2)*100,"%"))
```

```
## # A tibble: 8 x 3
##   categories       democrat props
##   <chr>               <dbl> <chr>
## 1 Somewhat Oppose         0 9%
## 2 Somewhat Oppose         1 29%
## 3 Somewhat Support        0 38%
## 4 Somewhat Support        1 14%
## 5 Strongly Oppose         0 11%
## 6 Strongly Oppose         1 53%
## 7 Strongly Support        0 41%
## 8 Strongly Support        1 4%
```

We see the same general pattern that we did in the ordered probit example. Moreover, the general magnitude of the predicted effects are the same. This is confirmatory that despite how we conceptualized this model—whether we thought the categories are ordered or not—we still get similar results.

Simulating Confidence Intervals

One issue with the above predictions is that we have no sense of how certain we are about these predictions. As we did with binary and ordered response models, we can estimated our uncertainty around our predictions via monte carlo simulation.

Below I'll use the `obsval` package to calculate the 95% confidence interval around the predicted effects. The process of doing this manually is a little involved as we saw last time with binary responses, but it follows the same basic steps (try doing it for yourself!)

```
require(obsval) # Load the package

# Re-estimate the model using obsval
multi_mod2 <-
  obsval(warsup ~ dem + female,
         data = dat,
         ci = .95,
         n.draws = 1000,
         baseline.category = "strongly oppose", # need to spec baseline
         reg.model = "mlogit",
         effect.var = "dem",
         effect.vals = c(0,1))
```

```
## # weights:  16 (9 variable)
## initial  value 693.147181
## iter  10 value 578.220271
## final  value 571.563301
## converged
```

```r
# Check the model summary to make sure everything looks right
summary(multi_mod2$model)
```

```
## Call:
## multinom(formula = fmla, data = data, Hess = TRUE)
##
## Coefficients:
##                   (Intercept)        dem       female
## somewhat oppose    0.04525367 -0.4162225 -0.407679600
## somewhat support   1.20336368 -2.5008800  0.007914799
## strongly support   1.59309190 -3.8175076 -0.550507756
##
## Std. Errors:
##                   (Intercept)       dem    female
## somewhat oppose     0.3199384 0.3245411 0.2612472
## somewhat support    0.2735472 0.2888118 0.2761655
## strongly support    0.2732472 0.3842137 0.3056970
##
## Residual Deviance: 1143.127
## AIC: 1161.127
```

Let's visualize the predicted probabilities using the simulated intervals. Note that like the ordered models, the predicted probabilities are stored as arrays: 1000 simulations by 4 coefficient values by two manipulation states (democrat or not).

```r
dim(multi_mod2$preds)
```

```
## [1] 1000    4    2
```

We can unpack these simulated distributions, summarize them (mean and 95% confidence intervale) and plot them.

```r
# Function extracts the 2.5%, mean, and 97.5% interval
extract = function(x) c(quantile(x,.025),ave_pred=mean(x),quantile(x,.975))

# Apply to systematically draw it
non_dem = apply(multi_mod2$preds[,,1],2,extract)
dem = apply(multi_mod2$preds[,,2],2,extract)
```

Reformat as a table.

```r
non_dem = data.frame(t(non_dem)) %>%
  rownames_to_column("category") %>%
  mutate(democrat = 0)
dem = data.frame(t(dem)) %>%
  rownames_to_column("category") %>%
  mutate(democrat = 1)
```

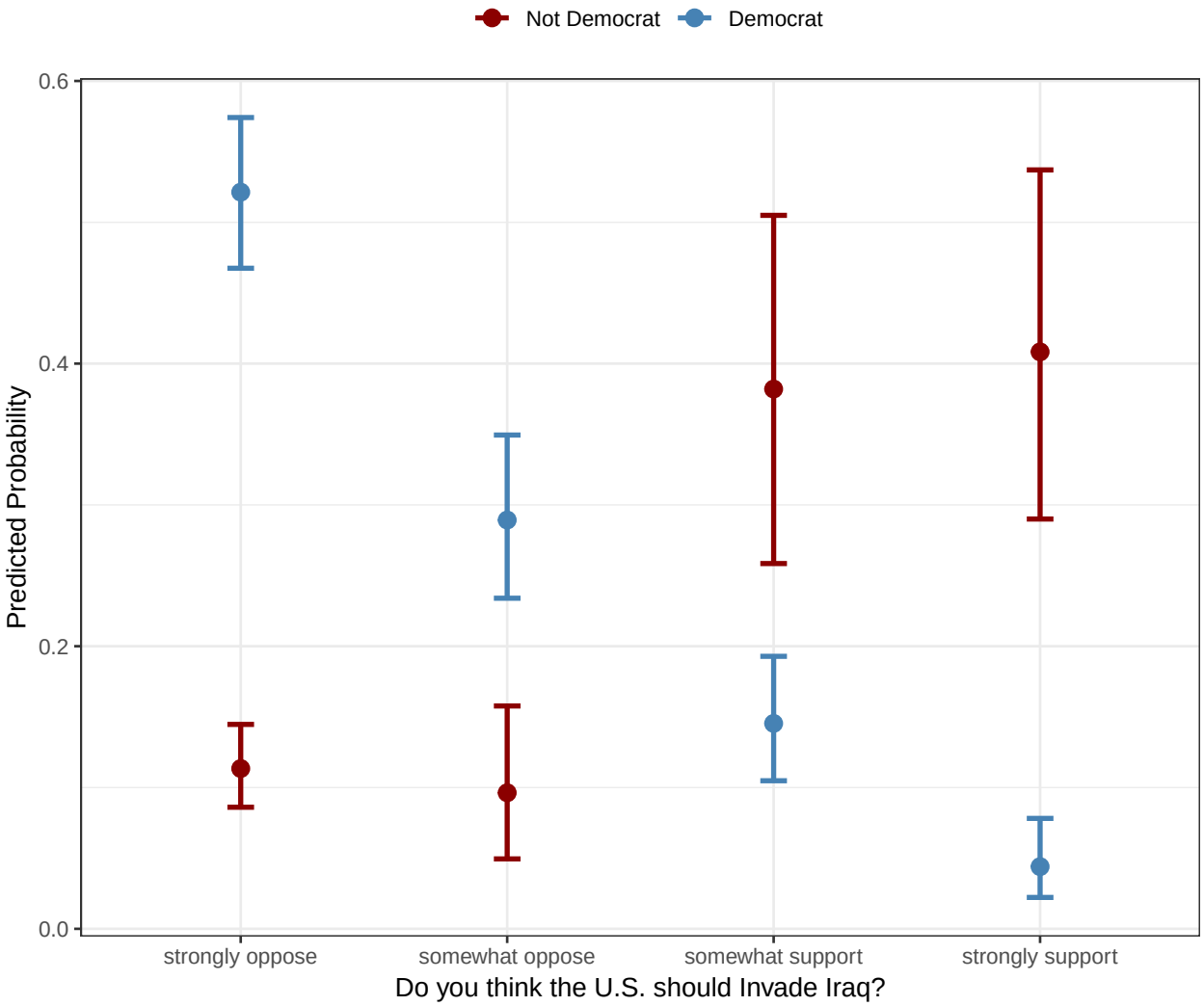
```
predicted_effects = bind_rows(dem,non_dem)
predicted_effects
```

```
##           category      X2.5.   ave_pred     X97.5. democrat
## 1  somewhat oppose 0.23395624 0.28929165 0.34944798        1
## 2 somewhat support 0.10479382 0.14536741 0.19285107        1
## 3 strongly support 0.02221104 0.04401585 0.07815265        1
## 4  strongly oppose 0.46747801 0.52132509 0.57412711        1
## 5  somewhat oppose 0.04948838 0.09629077 0.15770985        0
## 6 somewhat support 0.25850464 0.38199491 0.50491918        0
## 7 strongly support 0.29000900 0.40829038 0.53707379        0
## 8  strongly oppose 0.08604159 0.11342394 0.14464812        0
```

Finally, let's visualize![3]

```
predicted_effects %>%
  mutate(category = factor(category,levels = tolower(categories))) %>%
  ggplot(aes(category,ave_pred,color=factor(democrat))) +
  geom_point(size=3) +
  geom_errorbar(aes(x=category,ymin=`X2.5.`,ymax=`X97.5.`),
                    width=.1,size=1) +
  labs(y="Predicted Probability",
       x="Do you think the U.S. should Invade Iraq?",
       color="") +
  scale_color_manual(values=c("darkred","steelblue"),
                     labels=c("Not Democrat","Democrat")) +
  theme_bw() +
  theme(legend.position = "top")
```

[3]Note that I'm playing with the factor levels in the below plot to ensure the right ordering when visualizing.



We see the same general trend in the predicted effects across the four choices as we did in the ordered probit example; however, with a one notable difference, the confidence intervals around the non-dem support estimates are larger. Given that we're essentially estimating a series of logit models with coefficients for each choice, the larger confidence intervals may reflect the increase in the number of parameters that we have. More importantly, there is more uncertainty in the model as we have uncertainty estimates around each coefficient, which is being incorporated into the end estimate. if anything, this reinforces the need to estimate intervals around our predicted effects.

Another difference between the ordered and multinomial results is the distance in the predicted effect between democrats and non-democrats in the somewhat oppose choice. In the ordered probit, these intervals were far closer, whereas in the multinomial model they're further apart.

Finally, keep in the mind that we used an ordered *probit* model and we are comparing it to a multinomial *logistic* model. As we can see, the differences in link functions don't really change the story, but it's always important to keep in mind when comparing results.