Lab 8: Binomiaal Models in R

TA: Isabel Laterzo

3/17/2021

Today's lab has been adapted from code by Chelsea Estancona and Simon Hoellerbauer.

Binomial Models

Today we'll work through a fairly standard problem: you have a dichotomous dependent variable and a set of independent variables, and you want to estimate a model. Let's walk through a full example.

Load the `Lab_8_data.dta` file. These are data from a previous project (Chelsea Estancona ('18)'s) about paramilitary groups in Colombia. Our DV is a dichotomous variable coding paramilitary presence in certain municipalities.

```
colombia <- read_dta('Lab_8_data.dta')
```

Let's plot our DV. This is always a good practice for *all* variables (independent or dependent) in your work.

```
library(ggplot2)
ggplot(colombia, aes(x = as.factor(AUC))) +
  geom_bar() +
  labs(x = 'AUC Presence') +
  theme_bw()
```

Note that plotting in this way can also give us information about missingness (which there's a lot of on this variable). Remember - what does R do with our missing data if we leave it to handle it on its own?

Now we're ready to put together a simple model.

```
#for a logit model, how should we specify family?

m1 <- glm(AUC ~ tribut_2_log + tribut_2_log_sq + pop_attacks + disbogota +
            drugs + pop_log, data=colombia, family = __________)

summary(m1)
```

If we wanted to use a probit link:

```
#for a probit model, how should we specify family?

m2 <- glm(AUC ~ ., data = colombia, family = __________)
#what is the . doing above?
summary(m2)
```

The sign (and generally, significance) of our coefficient estimates remain the same. The values differ due to these link functions.

This is all very well and good, but how useful are the coefficient estimates? How do we interpret them?

We might want to use our model to estimate predicted probabilities, as this is a readily-understood measure of how changes in our independent variable affect the probability of observing a certain outcome. In doing so, we'll want to think about a 'hypothetical case' for which we want to predict: for example, here, we'll calculate the predicted probability of the United Self-Defenses of Colombia or Autodefensas Unidas de Colombia (AUC) presence in a municipality with the mean tax income, mean attacks on population, mean distance from the capitol, drug crops, and the average population. *Note*: This is the so-called average case approach.

```
## calculate value of the linear predictor for this variable profile
linpred <- as.numeric(with(colombia, coef(m1)[1] +
                           coef(m1)[2] * mean(________, na.rm = T) +
                           coef(m1)[3] * mean(_________, na.rm = T)^2 +
                           ______________________________________
                           ______________________________________
                           ______________ #note coef 6 is unique
                           ______________________________________))

library(arm) # inverse logit function
#let's take a look!
invlogit #no parentheses

## invlogit lets us transform our log-odds to the range (0,1) - now interpretable
## as a standard probability
invlogit(linpred)
```

We can also do this using the 'predict' function:

```
## create a 'new data' frame: we want to predict the probability for a hypothetical situation.
new_data <- with(colombia, data.frame(tribut_2_log = mean(tribut_2_log, na.rm = T),
                                      tribut_2_log_sq = mean(tribut_2_log, na.rm = T)^2,
                                      pop_attacks = mean(pop_attacks, na.rm = T),
                                      disbogota = mean(disbogota, na.rm = T),
                                      drugs = c(0, 1),
                                      pop_log = mean(pop_log, na.rm = T)))
## note above that I include both possible values for drugs. the 'data.frame()'
## command is flexible and will intelligently duplicate the other values

#Here use predict()
predict(_____, newdata = ________, type = _________)
```

What if we were interested in a *range* of values' predicted probability? We could modify our newdata frame and our predict() function to allow for this.

```
min_tax <- min(colombia$tribut_2_log, na.rm = T)
max_tax <- max(colombia$tribut_2_log, na.rm = T)
min_tax_2 <- min(colombia$tribut_2_log_sq, na.rm = T)
max_tax_2 <- max(colombia$tribut_2_log_sq, na.rm = T)

#for moddel 1 - logit
new_data_2 <- with(colombia,
                   data.frame(tribut_2_log = seq(________, _________,
                                                 length.out = 100),
                              tribut_2_log_sq = seq(________, _________,
                                                    length.out = 100),
                              pop_attacks = _________,
                              disbogota = _________,
                              drugs = 1,
```

```
                            pop_log = _________))

#for moddel 1 - probit
new_data_3 <- #Same as above?


predict(m1, newdata = new_data_2, type = 'response') # what is the length of this output?
predict(m2, newdata = new_data_3, type = 'response')

#we can calculate the logit predicted probs by hand by doing 1/(1 + exp(-linpred))
#How could we calculate this for probit?

#We can use pnorm! (this is, after all, the difference between the logit and probit)
all(predict(m2, newdata = new_data_3, type = "response") ==
      pnorm(as.matrix(cbind(1, new_data_3)) %*% coef(m2)))
```

Note that in this case, it is perfectly ok to vary the tax variable and the the squared tax value from its min to its max, as the square roots of the minimum and maximum squared are equal to the minimum and maximum (unless for some reason you have missingness in one variable at the max and min, which should be unlikely).

We know from the numeric output that logit and probit links produce different coefficient estimates, but how do these differences translate into predicted probabilities? We can plot the predicted probabilities from each model to quickly inspect these differences.

```
## generate predicted probabilities for municipality with drugs and all other
## variables at their means
pred_output <- data.frame(Logit = predict(_____, newdata = _______, type = _______),
                          Probit = predict(____, newdata = ________, type = ______),
                          tax = new_data_2$tribut_2_log)

plot_pred1 <- gather(pred_output, key = "link", value = "Pred_Probs", -tax)

## plot predicted probabilities
ggplot(data = plot_pred1, aes(x = tax, y = Pred_Probs, color = link)) +
  geom_line() +
  scale_color_discrete(name = 'Link Function') +
  labs(x = 'Tax Revenue', y = 'Predicted Probability') +
  theme_bw()
```

Simulation

Simulation strikes again! This time around, we will do similar things, but with non-Normal outcomes, where simulations are even more helpful.

Today, we're going to simulate predicted probabilities for a model with an interaction term and plot our results. We will use both the average case and the observed-case approaches.

Using Average-Case Approach

The following should all look familiar:

```
## number of simulations for predicted probabilities
N_sim <- 1000

## extract beta hats and construct vcv from our first model
```

```
betas <- ______
vcv <- _______

## use the mvrnorm function with the betas, vcv, and # of simulations to simulate
## parameter estimates. will have 1,000 estimates of each beta
beta_sims <- mvrnorm(N_sim, betas, vcv)

## create a vector that is a sequence of 100 points, evenly spaced between the
## lowest and highest observed values of tax income in the data set. we have to
## do this for the (tax^2) variable, too!
tax_min <- with(colombia, min(tribut_2_log, na.rm = T))
tax_max <- with(colombia, max(tribut_2_log, na.rm = T))
tax_vec <- seq(from = tax_min, to = tax_max, length = 100)
trans_vec <- (tax_vec^2)
## these will be the x values over which we can plot the predicted probability.
## we're interested in how the predicted probability of AUC presence varies over
## the municipalities' values of tax revenue

## create a hypothetical independent variable as before: mean attacks on population,
## mean distance from the capitol, drug crops set to 1, and the average population.
hyp <- with(colombia, data.frame(intercept = 1,
                                 tribut_2_log = tax_vec,
                                 tribut_2_log_sq = trans_vec,
                                 pop_attacks = mean(pop_attacks, na.rm = T),
                                 disbogota = mean(disbogota, na.rm = T),
                                 drugs = 1,
                                 pop_log = mean(pop_log, na.rm = T)))

#We could also have done this using model.matrix()

## create an empty matrix of values to fill with simulated predicted probabilities
pp_sims <- matrix(NA, nrow = _____, ncol = ________)
#Using a for() loop to fill wth simulated pred. probs
for(i in 1:N_sim) {

  pp_sims[i, ] <- invlogit(as.matrix(hyp) %*% beta_sims[i, ])

}

#we could also do this with less code:
pp_sims2 <- invlogit(as.matrix(hyp) %*% t(beta_sims))

dim(pp_sims)
dim(pp_sims2)

all(t(pp_sims) == pp_sims2)
```

We're not going to worry about confidence intervals today, but below is the code for them as well.

```
pe <- apply(pp_sims, 2, mean)
#lo <- apply(pp_sims, 2, quantile, prob = .025)
#hi <- apply(pp_sims, 2, quantile, prob = .975)

sim_output <- data.frame(tax = hyp[, 2], pe)
```

```
ggplot(sim_output, aes(x = tax, y = pe)) +
  geom_line(aes(color = 'Predicted Probability Over Values of Municipal Revenue'),
            size = 1) +
  labs(x = 'Revenue', y = 'Probability of Paramilitary Presence') +
  theme_bw() +
  theme(legend.title = element_text(), legend.position = 'bottom',
        panel.grid.minor.x = element_blank(),
        panel.grid.minor.y = element_blank()) +
  scale_color_manual(values = 'black', name = '')
```

Using Observed-Case Approach - Average Predicted Effects

The code above created a hypothetical set of cases in which we might be interested, but what if we wanted more of an average effect? To this end, we can code and plot an 'average predictive effect' that covers the values of our key variable (we still generate predicted probabilities 'along' these values) and all possible values of the other variables in our dataset.

```
## create a 'temporary' dataset - bind a 1 for the intercept and omit missingness
temporary <- model.matrix( _____________, data = colombia)

## create an object and store predicted probabilities
## look into the ?map_dfc function from the purrr package (part of the tidyverse)

#note this will take like a minute
pp <- purrr::map_dfc(tax_vec, function(j) {
    temporary[, "tribut_2_log"] <- j
    temporary[, "tribut_2_log_sq"] <- j^2

    pp <- invlogit(temporary %*% t(beta_sims))

    #getting mean for each set of betas
    pp <- apply(pp, 2, mean)

    return(pp)
    })

## find the mean (average predictive effect) for each observation
plotdat <- apply(pp, ___, _____)

# add x vector and convert to data frame
plotdat <- data.frame(tax_vec, plotdat)

# better names and show the first few rows (to check)
colnames(plotdat) <- c('tax', 'pe')
head(plotdat)

#now let's rowbind in the average-case predicted probabilities, so that we can
#compare them
plotdat <- rbind(plotdat, sim_output) %>%
  dplyr::mutate(Approach = c(rep("Observed-Case",length(tax_vec)),
                      rep("Average-Case", length(tax_vec))))

## plot of predicted probabilities
```

```
ggplot(plotdat, aes(x = tax, y = pe, color = Approach,
                    fill = Approach)) +
  geom_line() +
  labs(x = 'Tax Revenue', y = 'Predicted Probability') +
  ylim(c(0, 1)) +
  theme_bw()
```

Assignment

Install and load the package 'Zelig.' Call the data 'turnout.' Fit a logit model that predicts voting based on race, education and income.

Now, fit a second model that predicts voting based on race, education, income, and age. Assume that you've hypothesized that age has a curvilinear relationship with voting turnout, such that lower ages and higher ages are less likely to vote. Include this transformed variable.

Compare the model fit of these two options. Which do you choose?

Simulate and plot predicted probabilities over the range of age, holding all other variables at their mean, for the second model. Plot this.

Generate an average predicted effect over the observed values of income for the first model. Plot this.