POLS6382 Quantitative Research Methods III: Maximum Likelihood Estimation

Ling Zhu
Department of Political Science
University of Houston
September 27, 2021

Lab 4: Binary Response Models: Advanced Topics

Objectives

- Learn how to estimate a heterokedastic probit model.
- Learn how to estimate a rate-event logit model.
- Learn how to use post-estimation simulation to consider uncertainty (of parameter estimation) in predication calculations.

1 Comparing Standard and Heteroskedastic Probit Model with Simulated Data

When heterokedasdicity occurs, a standard probit model will produce inconsistent and inefficient estimation, thus we need to consider a heteroskedastic probit model as the alternative. Function `hetglm()`from package `glmx` can be used to fit a heteroskedastic probit model. In this section, we'll use a simulated dataset to show why a standard probit model is problematic when the homoskedasticity assumption is violated. We'll also compare estimation results between a standard and a heteroskedastic probit model.

1.1 Simulate the Example Datafile

We simulate a dataset by the following steps.

———————————- R Code———————————-

```
set.seed(48)
n<- 500
x<- rnorm(n)
ystar<-1+x+rnorm(n, sd = exp(x))
y<-ifelse(ystar>0, 1, 0)
simdata<-data.frame(cbind(y,x))
```

1. Define sample size: n=500.
2. Randomly draw 500 observations for variable y from a standard normal distribution.
3. Define the latent continuous scale of $ystar$, as such, $ystar = 1 + x + \epsilon$. ϵ is drawn from a normal distribution with mean equals to 0 and standard deviation equals to $exp(x)$. Defining $sd = exp(x)$, we let the variance of ϵ vary along x, hence, ϵ^2 is not constant.
4. Simulate observed binary variable y based on the latent scale $ystar$. `if else()` is used to define the classification rule: classify the observation to be 1 if$ystar > 0$, and 0 otherwise.
5. Make a data frame by combining variable y and x.

———————————- R Code———————————-

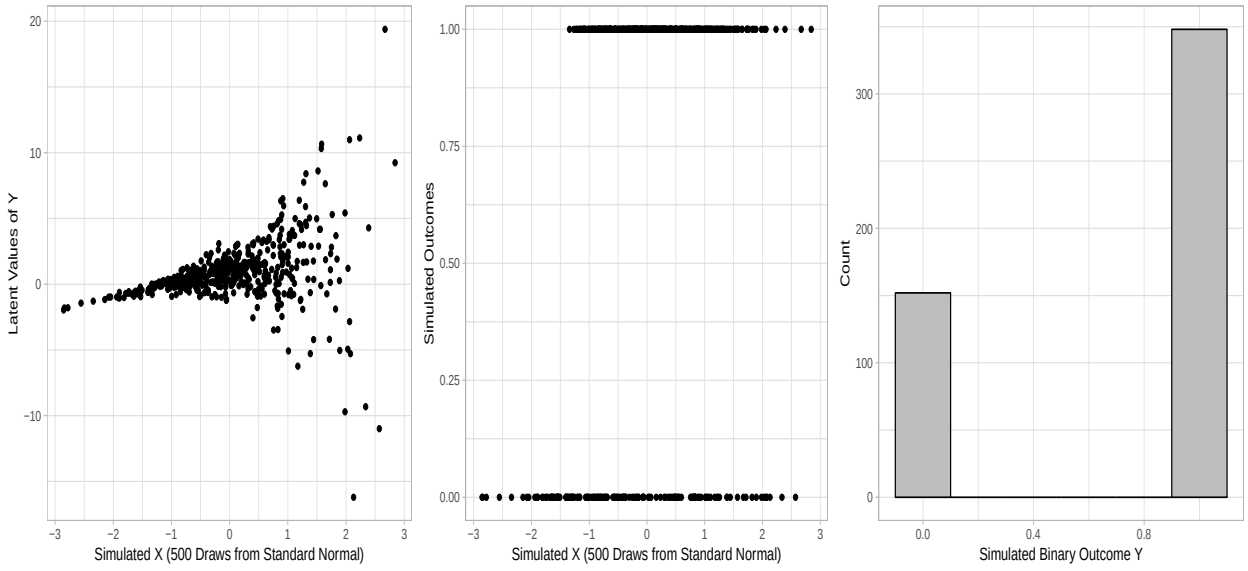
```
visual1<-ggplot(data=simdata)+
  geom_point(aes(x=x, y=ystar))+
  labs(x="Simulated X (500 Draws from Standard Normal)",
       y="Latent Values of Y")+
  theme_light()

visual2<-ggplot(data=simdata)+
  geom_point(aes(x=x, y=y))+
  labs(x="Simulated X (500 Draws from Standard Normal)",
       y="Simulated Outcomes")+
  theme_light()

visual3<-ggplot(data=simdata)+
  geom_histogram(aes(x=y), binwidth = 0.2, color="black", fill="gray")+
  labs(x="Simulated Binary Outcome Y",
       y="Count")+
  theme_light()

require(ggpubr)
pdf(file="simdata.pdf", height=5, width=15)
ggarrange(visual1,visual2, visual3, ncol=3, nrow=1)
dev.off()
```

- R Output -



Visualizing the simulated data, we see that y^* has non-constant error variance. By our simulation, we classify 350 cases with 1s, and 150 cases with 0s.

1.2 Estimating a Standard Probit Model When Variance Is Not Constant

What would happen if we estimate a standard probit model when the variance is not constant? It will produce biased results. Table 1 reports two coefficients: intercept=0.521, the slope coefficient of x is 0.345.

Both numbers deviate from 1 (the true values) substantially (i.e. biased).

———————————- R Code———————————-

```
model1 <- glm(y ~ x, family = binomial(link = "probit"),data=simdata)
```

———————————- R Output———————————-

Table 1: Standard Probit Model

	Dependent variable:
	y
x	0.345^{***}
	(0.064)
Constant	0.521^{***}
	(0.060)
Observations	500
Log Likelihood	−290.805
Akaike Inf. Crit.	585.610
Note:	*p<0.1; **p<0.05; ***p<0.01

1.3 Heteroskedastic Probit Model

A more approperiate specification is the heteroskedastic probit model, using which we can specify a scale parameter to model the non-constant variance term. The syntax of `hetglm()` is as the following. We define the variance as a linear function of x (recall that we set the variance to have a mean of 0, but sd is defined as a function of x).

———————————- R Code———————————-

```
model2 <- hetglm(y ~ x | x, family=binomial(link="probit"),data=simdata)
```

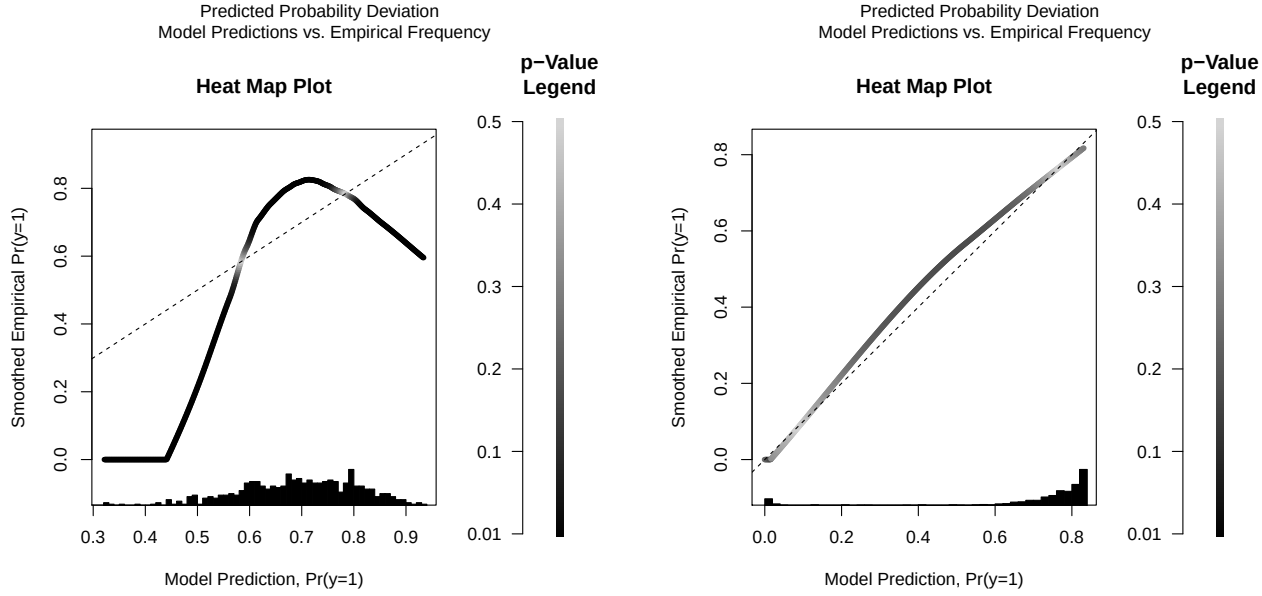
———————————- R Output———————————-

Table 2: Heteroskedastic Probit Model

	Dependent variable:
	y
x	0.881^{***}
	(0.081)
Constant	0.954^{***}
	(0.081)
Latent Scale Model Coefficeints	
Estimated Sd.	0.973^{***}
	(0.112)
Observations	500
Log Likelihood	−252.188
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 2 shows the heteroskedastic probit model produces coefficients that are much closer to the true intercept (1) and true slope (1). The estimated latent scale variance parameter has a positive coefficient. This means that the variance of y^* increases as x increases, which is just as what we defined. Figure 1 compares the two specifications using heat map fit plot. Figure 1-(1) clearly shows that Model 1 is a mis-specified model. Model 2 substantially improves the model fit.

Figure 1: Comparing Two Specifications



(a) Model 1

(b) Model 2

2 Estimate a Heteroskedastic Probit Model with Real Data

In this section, we use a real data example to compare standard and heteroskedastic probit model. We use the labor force participation data example from package`AER`. The dataset contains 753 observations and 21 variables. The variable of interest is a binary variable `participation`, which is coded as 1 if an individual is in labor force, and 0 otherwise.

2.1 Estimation with `heterglm`

The first model equation is a standard probit model. The second model equation estimates the same standard probit using `heterglm()`, by setting the latent scale to be 1. Doing so, we will have the same number of parameters for this model and the heteroskedastic probit model (equation 3). We can perform likelihood ratio test to compare two models.

———————————- R Code———————————-

```
data("PSID1976", package = "AER")
PSID1976$kids <- with(PSID1976, factor((youngkids + oldkids) > 0,
                levels = c(FALSE, TRUE), labels = c("no", "yes")))
PSID1976$fincome <- PSID1976$fincome/10000

#Standard probit model via glm()
labormodel1<- glm(participation ~ age + I(age^2) +
              fincome + education + kids,
             data = PSID1976, family = binomial(link = "probit"))

# Standard probit model via hetglm() with constant scale
labormodel2 <- hetglm(participation ~ age + I(age^2) +
                fincome + education + kids | 1,
                data = PSID1976)

#Heteroskedastic Probit model with varying scale
heterlabor<- hetglm(participation ~ age + I(age^2) +
              fincome + education + kids | kids + fincome,
               data = PSID1976)
```

———————————- Output———————————-

Table 3: Determinants of Labor Force Participation: Heteroskedastic Probit Model

	Dependent variable:
	participation
age	0.264**
	(0.125)
I(age^2)	−0.004**
	(0.002)
fincome	0.424*
	(0.218)
education	0.140**
	(0.057)
kidsyes	−0.879***
	(0.312)
Constant	−6.030**
	(2.740)
Latent scale coefficients	
Estimate Std.	
kidsyes	-0.141
	(0.289)
fincome	0.313**
	(0.118)
Observations	753
Log Likelihood	−487.636
Note:	*p<0.1; **p<0.05; ***p<0.01

2.2 Compare Two Specifications

We use the likelihood ratio test to compare two model specifications. The heteroskedastic probit model is preferred, but it only slightly improves model fit. Check prediction results using a "confusion table", we see that the heteroskedastic probit model outperforms the standard model, but increasing the number of correct classifications only by 8.

———————————- R Code/Output———————————-

```
# Likelihood ratio test
lrtest(labormodel2, heterlabor)
Likelihood ratio test

Model 1: participation ~ age + I(age^2) + fincome + education + kids |
    1
Model 2: participation ~ age + I(age^2) + fincome + education + kids |
    kids + fincome
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   6 -490.85
```

```
2   8 -487.64  2 6.4245    0.04027 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Check for predictions
table(true = PSID1976$participation,
+ predicted = fitted(labormodel1) <= 0.5)
     predicted
true  FALSE TRUE
  no    219  106
  yes   357   71
 table(true = PSID1976$participation,
+ predicted = fitted(heterlabor) <= 0.5)
     predicted
true  FALSE TRUE
  no    210  115
  yes   358   70
```

---

3 Post Estimation Simulations with Zelig

3.1 What Is Zelig and What Does Zelig Do?

`Zelig` is an R package developed by Kosuke Imai, Gary King, and Olivia Lau. It is a common framework for estimating a variety of statistical models and integrates simulation-based methods (e.g. bootstrapping, Bayeisan inference, etc.) to improve model performance and the accuracy of predictions. A similar (but less powerful) `Stata` module developed by Gary King and his associates is `Clarify`. For more detailed discussions on `Zelig` and `Clarify`, see the following two articles.

1. King, Gary, Michael Tomz, and Jason Wittenberg. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." *American Journal of Political Science* 44(2):347-361.
2. Kosuke Imai, Gary King, and Olivia Lau. "Toward A Common Framework for Statistical Analysis and Development." *Journal of Computational and Graphical Statistics* 17(4): 892-913.

`Zelig` can do a variety of things: simulating scientific quantities of interest, performing point and uncertainty estimation, multiple imputation, matching methods, counterfactual evaluations, replication, etc. It also supports a large number of statistical models: Bayesian and frequentist models, multiple equation models, times-series-cross-section models, multi-level models, etc.

3.2 Point/Uncertainty Estimates and Probability Calculation

In this section, we use the `turnout` example from `Zelig` to show how one can use `Zelig` to estimate a binary response model, then use simulation-based methods to perform out-sample predictions. This is also one of the examples used in King et. al AJPS 2000 paper regarding `Clarify`.

Road map:

1. Estimate the model using function `zelig()`.

2. Sett specific values for explanatory variables using function `setx()`.
3. Compute quantity of interest using function `sim()`. `Zelig` supports simulations for expected values given the model and x, predicted values given by the fitted values, first differences, risk ratios, average predicted and expected treatment effects.

——————————- R Code——————————————————-

```
data(turnout,package="Zelig")
z.out<-zelig(vote~race+educate+age+I(age^2)+income,model="logit",data=turnout)
```

——————————R Output——————————————————-

Table 4: Determinants of Turnout: Logit Regression Model

	Dependent variable: Turnout
racewhite	0.272^{*}
	(0.147)
educate	0.176^{***}
	(0.020)
age	0.095^{***}
	(0.018)
age^2	-0.001^{***}
	(0.0002)
income	0.152^{***}
	(0.028)
Constant	-4.342^{***}
	(0.476)
Observations	2,000
Log Likelihood	−1,004.787
Akaike Inf. Crit.	2,021.575
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 4 shows we specify the probability of turnout is a function of individuals' race, education, age and income. In the model, we also add a square term of *age* to depict the curvy-linear relationship between age and the probability of turnout. We find that both age and education significantly affect the probability of turnout. After estimating the logit model, we are interested in calculating the predicted probabilities of *turnout* along *age* and *education*. We do so by setting the values of *age* and *education*.The first statement in `sets()` is the model object. The second statement "`education=12`" sets year of education to be 12. "`age=18:95`" let values for *age* change across its full range. Note that we did not specify the values for *income* and age^2. `setx()` will automatically calculate age^2 based on the values we set for *age*. All remaining variables are hold at their means as the default. *setx*() also define the intercept to be 1 as the default.

——————————- R Code——————————————————-

```
x.low<-setx(z.out, educate=12, age=18:95)
x.high<-setx(z.out, educate=16, age=18:95)
```

——————————————————————————————-

The third step is to simulate the quantity of interest. `sim()` works along the following steps:

1. Simulate $\tilde{\beta}$ from asymptotic normal distribution. The default number of simulation is 1,000. You can also define the number of simulations by adding a statement, `n=...`.
2. Calculate quantity of interest: $\tilde{\pi}_i = \frac{1}{1+exp(-x_i\tilde{\beta})}$, for *eduation*= 12 and 16.

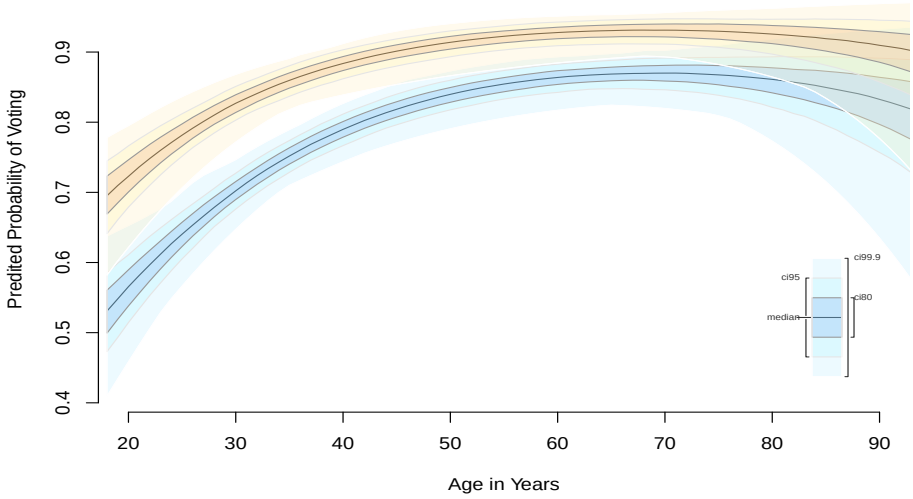
- R Code -

```
s.out<-sim(z.out, x=x.low, x1=x.high)
plot(s.out, xlab="Age in Years",
        ylab="Predited Probability of Voting",
        xlim=c(20,90))
```

R Output

Last, we use `plot()` to graph the predicted probabilities.

Figure 2: Zelig Example 1



4 Estimate a Rare-Event Logit Model with Zelig

`Zelig` also encompasses the rare-event logit regression model discussed in King and Zeng's two papers in *Political Analysis* and *International Organization*, respectively. Using `Zelig` to estimate a rare-event logit model is quite simple. In this section, we use the `mid` data example from `Zelig` to show the estimation process.

- R Code -

```
data(mid,package="Zelig")
reventmod<-zelig(conflict~major+ contig+power+maxdem+mindem+years, data=mid,
                 model="relogit",tau=1042/303772,
                 case.control="prior", bias.correct=TRUE)
```

R Output

Table 5: Determinants of Inter-State Military Conflict

	Dependent variable:
	conflict
major	2.433***
	(0.158)
contig	4.112***
	(0.158)
power	1.054***
	(0.217)
maxdem	0.048***
	(0.010)
mindem	−0.065***
	(0.013)
years	−0.063***
	(0.006)
Constant	−7.526***
	(0.180)
Observations	3,126
Akaike Inf. Crit.	1,882.5
Note:	*p<0.1; **p<0.05; ***p<0.01

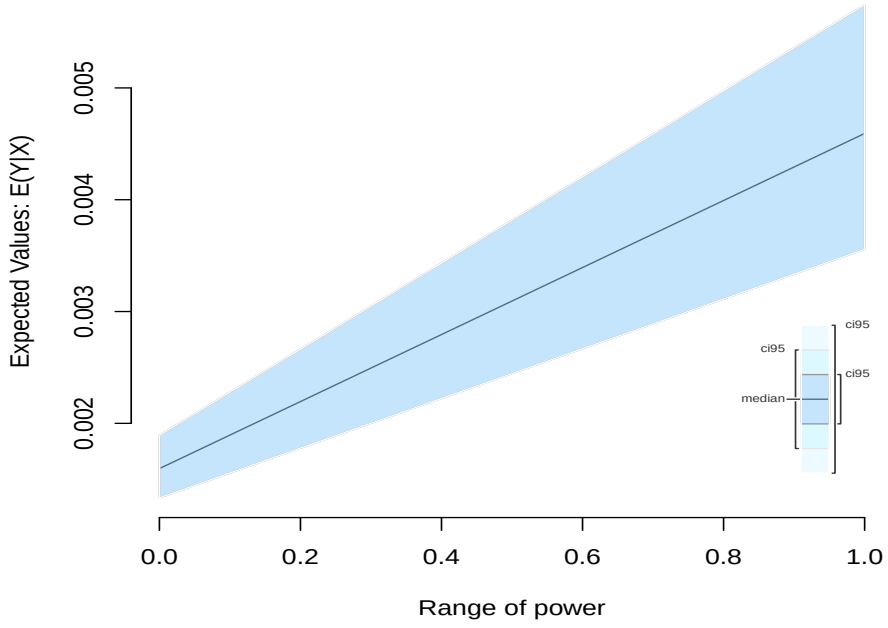
In this datafile, King et al. designed a 1:2 sample, including all cases of conflict, and about 2,000 cases of non-conflict. We get comparable results to what King and Zeng report in the Table 1 of their *IO* article. In our specification, we use the original sample proportion to define the value of τ. The method of bias correction is defined as "prior" (prior correction). If we write "weighting", then `Zelig` with just use the weighting method for bias correction. Similarly, we can use `setx()` and `sim()` to obtain/present the quantity of interest in a figure. Figure 3 is an example of how the probability of conflict changes as the value of *power* changes from 0 to 1.

———————————- R Code ————————————————————-

```
x.out1<-setx(reventmod,power=0:1)
#Simulate quantity of intersts
s.out1<-sim(reventmod,x=x.out1)
plot(s.out1,ci=95)
```

———————————- R Output ————————————————————-

Figure 3: Zelig Example 2: Rare Events Data



5 More on Post Estimation Simulations: With A Simulated Data Example

In this section, we will use another simulated data example to further illustrate how post-estimation simulations can help us to gauge uncertainty in coefficients. First, we simulate a dataset, in which we have 1,000 observations. In this dataset, we include two explanatory variables: X, coded as continuous and distributed as a standard normal distribution, and $Female$, coded as dichotomous. We define the latent continuous scale of y as $2X + 10Female + \epsilon$. We assume that the variance of ϵ is constant and normally distributed. The observed binary outcomes of y is simulated as 1s and 0s in this data example.

———————————- R Code———————————————————-

```
n <- 1000
simdata2<- data.frame(X = rnorm(n),
            Female = sample(c(0, 1), n, replace = T))
simdata2$Y <- with(simdata2, X * 2 + Female * 10 + rnorm(n, sd = 5))
simdata2$Y <- (simdata2$Y > 1) * 1
# Model:
myModel <- glm(Y ~ X + Female, data = simdata2, family = "binomial")
```

———————————- R Output———————————————————-

Table 6: Logit Regression: The Impact of X and Gender on Y

	Dependent variable:
	Y
X	0.678^{***}
	(0.095)
Female	3.442^{***}
	(0.232)
Constant	-0.315^{***}
	(0.095)
Observations	1,000
Log Likelihood	−411.503
Akaike Inf. Crit.	829.005
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 6 shows our basic specification, that the probability of observing $Y = 1$ is positively associated with both X and *Gender*. This can be generalized to any empirical examples, whereby we theorize that the binary dependent variable Y is predicted by a continuous variable X and dummy variable Z. When we concern about uncertainty in our estimated coefficients, we can use post-estimation simulations to improve the probability calculation. We do so by the following steps.

1. Define the number of simulations and some quantity of X and $Female$ for out-sample prediction. Here, we want 1,000 simulations. We also set the value range of X to be between the min and max values observed in our dataset. For variable $Female$, we just pick the two scenarios: $Female = 0$ and $Female = 1$.

```
------------------------- R Code -------------------------
nSims <- 1000
someScenarios <- expand.grid(1,  # Intercept
              seq(min(simdata2$X), max(simdata2$X), len = 100),
             c(0, 1))
----------------------------------------------------------
```

2. Randomly generate 1,000 draws of the parameter values for β_X and β_{Female}, for both the coefficients and variance. Because now we have more than one explanatory variables, instead of drawing values from a normal distribution, we need to get the 1,000 draws from a multivariate normal distribution. Note that by doing this simulation, we produce two sample distributions for our estimated coefficients β_X and β_{Female}, and two sample distributions for their associated variance σ^2_X and σ^2_{Female}. In each of these sample distributions, we have 1,000 cases. After the simulation, we calculate predicted probabilities of $Y = 1$ associated with each draw of β_X and β_{Female}, and the defined values of X and $Female$. Note that we define the scale of X to range between its min and max values, `length.out=100`. That means, for each simulated β_X and β_{Female}, we calculate 100 predicted probability values across the full range of X when $Female$=1, then 100 predicted probability values when $Female$=0. Because we simulated 1,000 draws (1,000 pairs of β_X and β_{Female}), we calculate 200,000 values in total.

```
------------------------- R Code -------------------------
simDraws <- mvrnorm(nSims, coef(myModel), vcov(myModel))
simYhats <- plogis(simDraws %*% t(someScenarios))
```

3. The next block of code makes a data frame including all these simulated probabilities, and put the datafile in long-format. This is the specific data format needed for using `ggplot2` to graph the simulation results. Finally, we use `ggplot2` to graph the simulated predicted probabilities (Pr(Y=1)) across the full range of variable X, but separating scenarios based on gender.

R Code

```
# Combine scenario definitions(gender id) and predictions.
predictionFrame <- data.frame(someScenarios, t(simYhats))
# Reshape wide -> long (for ggplot2)
longFrame <- melt(predictionFrame, id.vars = colnames(someScenarios))

p1<-ggplot(data = longFrame,
      aes(x = Var2, y = value, group = paste(variable, Var3),
     colour = factor(Var3)))
p1<-p1 + geom_line(alpha = I(1/sqrt(nSims)))
p1<-p1 + scale_x_continuous("Explanatory Variable X", expand = c(0, 0))
p1<-p1 + scale_y_continuous("Predicted Pr(Y=1)", limits = c(0, 1), expand = c(0, 0))
p1<-p1 + scale_colour_brewer(palette="Set1", labels = c("Male", "Female"))
p1<-p1 + guides(colour = guide_legend("Group ID",
    override.aes = list(alpha = 1)))  # Avoid an alpha-related legend problem
p1<-p1 + ggtitle("The Effect of X on Y by Gender Group")
p1<-p1 + theme_bw()
print(p1)  # This might take a few seconds...
```

R Output

Figure 4: Simulation Example

Figure 4 shows the effect of X on Pr(Y=1) for both gender groups. As our model shows, X has a positive

effect on Pr(Y=1), therefore, for both gender groups, we observe that Pr(Y=1) increases as X increases. $Female$ also has a positive coefficient, meaning that the Pr(Y=1) is expected to be greater when $Female$=0 than those when $Female$=0. In Figure 4, we do not just present one pair of predicted probabilities. Instead, we present 1,000 pairs! Each thin line in the "blue" and "red" thread represent one simulation. By presenting the full set of simulations, we visualize the uncertainty of the estimation. We see that we are very uncertain about the predicted probabilities when $X < -1$ and when $Female$=1. Similarly, we have low level of estimation certainty when $X > 1$ and $Female$=0.