POLS6382 Quantitative Method III
Maximum Likelihood Estimation

Lab 4: Binary Choice Models (2)

Ling Zhu and Emily Lee
Department of Political Science
University of Houston

2025/10/01

1. Learning Objectives

- Learn how to estimate a heteroskedastic probit model.
- Learn how to estimate a rare-event logit model.
- Learn how to use post-estimation simulation to consider uncertainty (of parameter estimation) in prediction calculations.

```
> rm(list=ls())
> setwd("/Users/lingzhu/Dropbox/UH Teaching/POLS6382_2025/2025 Labs/Lab 4")
> my_packages <- c("AER", "ggplot2", "glmx", "heatmapFit",
+                  "lmtest", "maxLik", "reshape2", "Zelig", "MASS","stats")
> invisible(lapply(my_packages, require, character.only = TRUE))
```

2. Comparing Standard and Heteroskedastic Probit Model with Simulated Data

2.1 Describing Variables

When heteroskedasticity occurs, a standard Probit model will produce inconsistent and inefficient estimation, thus we need to consider a heteroskedastic Probit model as the alternative. The function `hetglm()`from package `glmx` can be used to fit a heteroskedastic Probit model. In this section, we'll use a simulated data set 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.

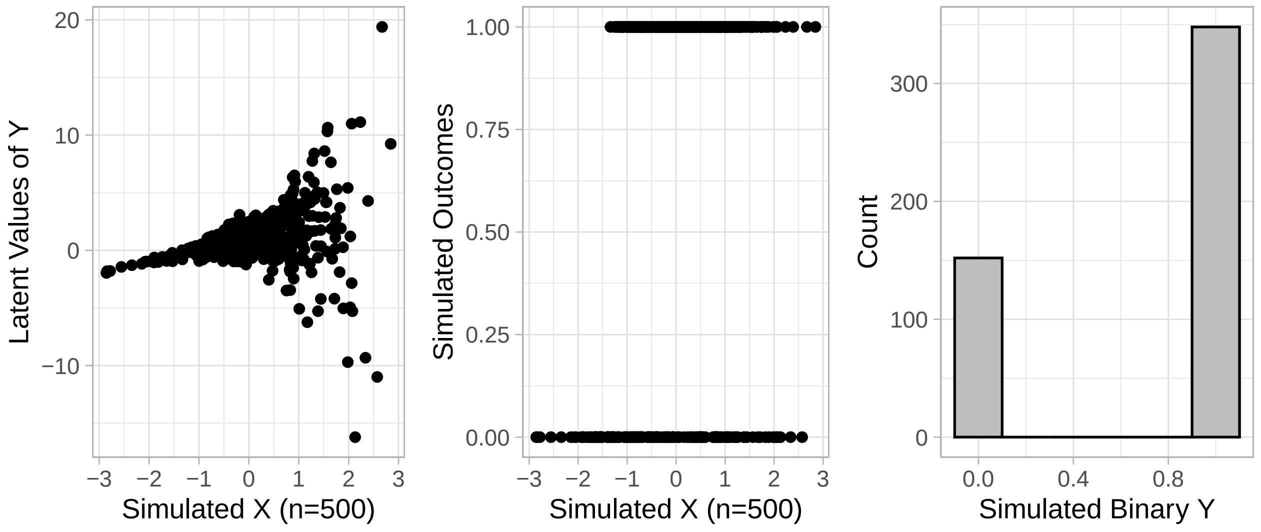
The data simulation process is specified as the following:

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 a mean of 0. The standard deviation is set to be $exp(x)$. 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 the variables y and x.

```
> 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))
```

We can visualize our simulated data, showing both the dependent variable (y) and the association between x and y. 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.

```
> visual1<-ggplot(data=simdata)+
+   geom_point(aes(x=x, y=ystar))+
+   labs(x="Simulated X (n=500)",
+         y="Latent Values of Y")+
+   theme_light()
>
> visual2<-ggplot(data=simdata)+
+   geom_point(aes(x=x, y=y))+
+   labs(x="Simulated X (n=500)",
+         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 Y",
+         y="Count")+
+   theme_light()
>
> require(ggpubr)
> ggarrange(visual1,visual2, visual3, ncol=3, nrow=1)
```



2.2 Estimating a Probit Model When Error 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. Model 1, shown below, 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).

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

```
Call:
glm(formula = y ~ x, family = binomial(link = "probit"), data = simdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2500  -1.1696   0.7154   0.8799   1.1373

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.52145    0.06029   8.649  < 2e-16 ***
x            0.34468    0.06403   5.383 7.32e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 614.22  on 499  degrees of freedom
Residual deviance: 581.61  on 498  degrees of freedom
AIC: 585.61

Number of Fisher Scoring iterations: 5
```

2.3 Heteroskedastic Probit Model

A more appropriate 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).

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

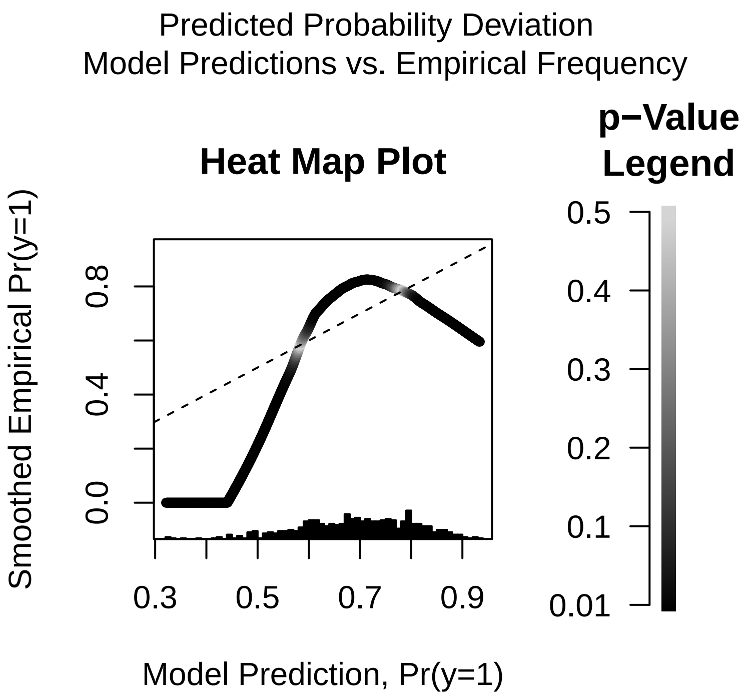
```
Call:
hetglm(formula = y ~ x | x, data = simdata, family = binomial(link = "probit"))

Deviance residuals:
    Min      1Q  Median      3Q     Max
-1.8838 -0.2589  0.6276  0.7310  1.7883

Coefficients (binomial model with probit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.95441    0.08093   11.79   <2e-16 ***
x            0.88140    0.08106   10.87   <2e-16 ***

Latent scale model coefficients (with log link):
  Estimate Std. Error z value Pr(>|z|)
x   0.9734     0.1116   8.723   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood: -252.2 on 3 Df
LR test for homoscedasticity: 77.23 on 1 Df, p-value: < 2.2e-16
Dispersion: 1
```

```
Number of iterations in nlminb optimization: 8
```

We observe that 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. The following figure compares the two specifications using heatmap fit plot. Figure 1-(1) clearly shows that Model 1 is a mis-specified model. Model 2 substantially improves the model fit.

```
> y<-simdata$y
> pred1<-predict(model1,type="response")
> pred2<-predict(model2,type="response")
> heatmap.fit(y,pred1,reps=1000,legend=FALSE)
```

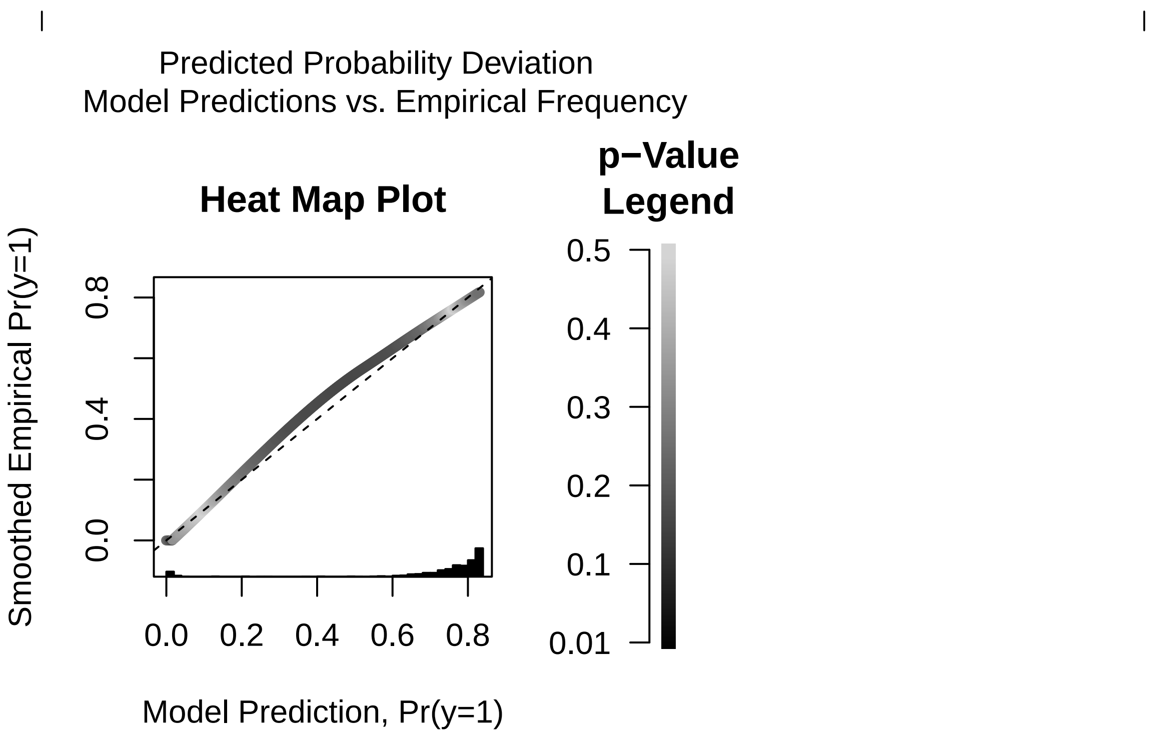
```
Calculating optimal loess bandwith...
aicc Chosen Span =  0.5788733

Generating Bootstrap Predictions...
  |                                                                            |
```



```
******************************************
80.8% of Observations have one-tailed p-value <= 0.1
Expected Maximum = 20%
******************************************
```

```
> heatmap.fit(y,pred2,reps=1000,legend=FALSE)
```

```
Calculating optimal loess bandwith...
aicc Chosen Span =  0.9776542

Generating Bootstrap Predictions...
```



```
*******************************************
0% of Observations have one-tailed p-value <= 0.1
Expected Maximum = 20%
*******************************************
```

3. 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`. This is also an empirical example discussed in Scott Long's book chapter. 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.

```
> load("PSID1976.rda")
> #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
>
> #Probit model using glm()
> labormodel1<- glm(participation ~ age + I(age^2) +
+                 fincome + education + kids,
+                 data = PSID1976, family = binomial(link = "probit"))
> summary(labormodel1)
```

```
Call:
glm(formula = participation ~ age + I(age^2) + fincome + education +
    kids, family = binomial(link = "probit"), data = PSID1976)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9205  -1.2261   0.7877   1.0634   1.6515

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.1568189  1.4040095  -2.961 0.003070 **
age          0.1853957  0.0662076   2.800 0.005107 **
I(age^2)    -0.0024259  0.0007762  -3.125 0.001775 **
fincome      0.0458029  0.0430557   1.064 0.287417
education    0.0981824  0.0228932   4.289  1.8e-05 ***
kidsyes     -0.4489872  0.1300252  -3.453 0.000554 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1029.7  on 752  degrees of freedom
Residual deviance:  981.7  on 747  degrees of freedom
AIC: 993.7

Number of Fisher Scoring iterations: 4
```

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

```
Call:
hetglm(formula = participation ~ age + I(age^2) + fincome + education +
    kids | 1, data = PSID1976)

Deviance residuals:
    Min      1Q  Median      3Q     Max
-1.9206 -1.2261  0.7877  1.0634  1.6515

Coefficients (binomial model with probit link):
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.1568063  1.4040077  -2.961 0.003070 **
age          0.1853951  0.0662075   2.800 0.005107 **
I(age^2)    -0.0024259  0.0007762  -3.125 0.001775 **
fincome      0.0458045  0.0430551   1.064 0.287393
education    0.0981823  0.0228932   4.289  1.8e-05 ***
kidsyes     -0.4489867  0.1300249  -3.453 0.000554 ***

Latent scale model coefficients (with log link):
None (constant scale = 1).
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood: -490.8 on 6 Df
Dispersion: 1
Number of iterations in nlminb optimization: 2
```

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

```
Call:
hetglm(formula = participation ~ age + I(age^2) + fincome + education +
    kids | kids + fincome, data = PSID1976)

Deviance residuals:
    Min      1Q  Median      3Q     Max
-1.8749 -1.2438  0.8045  1.0134  2.0913

Coefficients (binomial model with probit link):
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.029679   2.739767  -2.201  0.02775 *
age          0.264283   0.125208   2.111  0.03479 *
I(age^2)    -0.003628   0.001545  -2.349  0.01882 *
fincome      0.424431   0.217553   1.951  0.05107 .
education    0.140147   0.056767   2.469  0.01356 *
kidsyes     -0.879077   0.312269  -2.815  0.00488 **

Latent scale model coefficients (with log link):
        Estimate Std. Error z value Pr(>|z|)
kidsyes  -0.1408     0.2892  -0.487  0.62652
fincome   0.3129     0.1179   2.654  0.00795 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood: -487.6 on 8 Df
LR test for homoscedasticity: 6.425 on 2 Df, p-value: 0.04027
Dispersion: 1
Number of iterations in nlminb optimization: 13
```

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

```
> 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
```

4. Post Estimation Simulations with Zelig

4.1 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.

4.2 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:

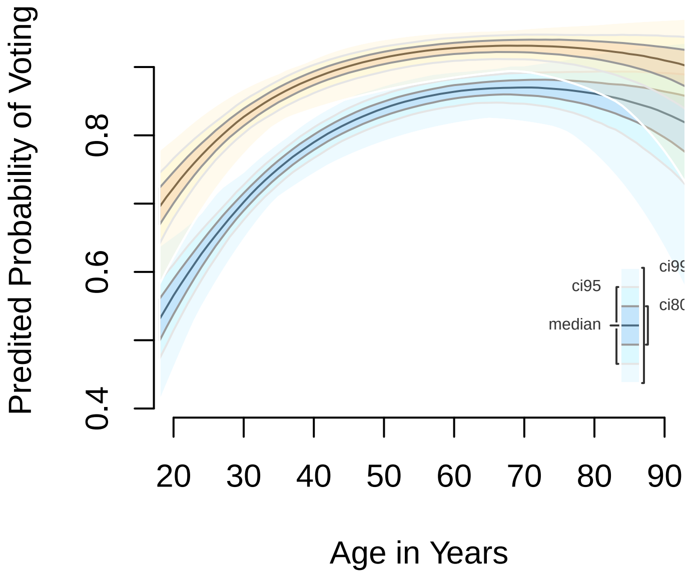
1. Estimate the model using function `zelig()`.
2. Sett specific values for explanatory variables using function `setx()`.
3. Compute quantity of interest using function `sim()`. `Zelig` supports simulations for expected values given the model and x, predicted values given by the fitted values, first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

We specify the probability of turnout as 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, `\texttt{education=12}" sets the 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 held at their means as the default. *setx*() also defines the intercept to be 1 as the default.

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

```
How to cite this model in Zelig:
  R Core Team. 2007.
  logit: Logistic Regression for Dichotomous Dependent Variables
  in Christine Choirat, Christopher Gandrud, James Honaker, Kosuke Imai, Gary King, and Olivia Lau,
  "Zelig: Everyone's Statistical Software," https://zeligproject.org/
```

```
> # Set explanatory variables
> x.low<-setx(z.out, educate=12, age=18:95)
> x.high<-setx(z.out, educate=16, age=18:95)
> #Simulate quantities of interest:
> 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))
```



5. Analyzing Rare Events Data with Zelig

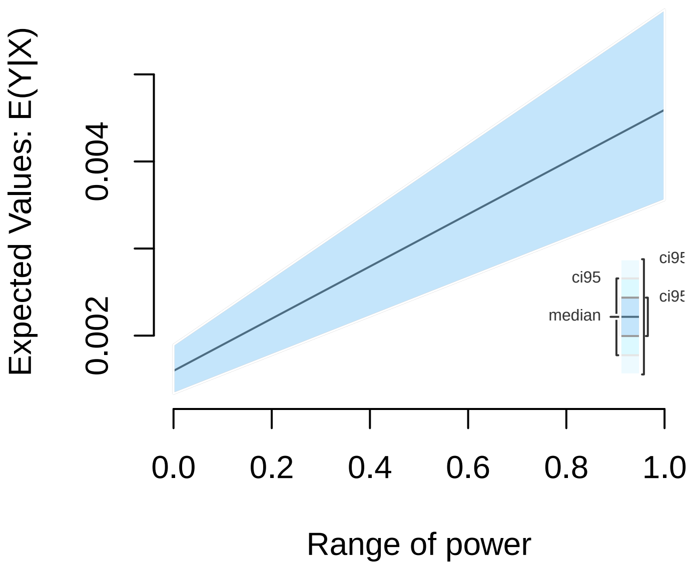
`Zelig` also encompasses the rare-event logit regression model discussed in King and Zeng's two papers in *Political Analysis* and *International Organization*, respectively. Using `Zelig` to estimate a rare-event logit model is quite simple. In this section, we use the `mid` data example from `Zelig`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

In this data file, 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 writeweighting`", 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. Appendix 1 includes R code to create different a cased-controlled sample (1:1) .

```
> 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)
```

```
How to cite this model in Zelig:
  Christine Choirat, Christopher Gandrud, James Honaker, Kosuke Imai, Gary King, and Olivia Lau. 2025.
  relogit: Rare Events Logistic Regression for Dichotomous Dependent Variables
  in Christine Choirat, Christopher Gandrud, James Honaker, Kosuke Imai, Gary King, and Olivia Lau,
  "Zelig: Everyone's Statistical Software," https://zeligproject.org/
```

```
> # summarize the model output
> summary(reventmod)

Model:

Call:
z5$zelig(formula = conflict ~ major + contig + power + maxdem +
    mindem + years, tau = 1042/303772, bias.correct = TRUE, case.control = "prior",
    data = mid)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0596  -0.0376  -0.0231   2.1085   4.4649

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.525688   0.179685 -41.883  < 2e-16
major        2.433432   0.157561  15.444  < 2e-16
contig       4.112491   0.157650  26.086  < 2e-16
power        1.053747   0.217243   4.851 1.23e-06
maxdem       0.048431   0.010065   4.812 1.50e-06
mindem      -0.065249   0.012802  -5.097 3.45e-07
years       -0.063359   0.005705 -11.106  < 2e-16

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3979.5  on 3125  degrees of freedom
Residual deviance: 1868.5  on 3119  degrees of freedom
AIC: 1882.5

Number of Fisher Scoring iterations: 6

Next step: Use 'setx' method
> # Let variable ``power" vary, and hold all other variables constant.
> x.out1<-setx(reventmod,power=0:1)
> #Simulate quantity of interests
> s.out1<-sim(reventmod,x=x.out1)
> plot(s.out1,ci=95)
```



6. Appendix 1: Code to Create a 1:1 Sample

```
> data(mid,package="Zelig")
> # create a subset only for cases if conflict=1
> subset1<-mid[which(mid$conflict=='1'),]
> # create a subset only for cases if conflict=0
> subset2<-mid[which(mid$conflict=='0'),]
> # creat a sample from subset2, randomly draw 1042 observations
> sample1<-subset2[sample(1:nrow(subset2),1042),]
> # create a new data frame, with a 1:1 sample design for 1s and 0s
> newdata<-rbind(subset1, sample1)
```

7. Appendix 2. More on simulations: visually weighted logit regression lines

In Appendix 2, we will use another simulated data example to further illustrate how post-estimation simulations can help us gauge uncertainty in coefficients. First, we simulate a data set, in which we have 1,000 observations. In this data set, 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 are simulated as 1s and 0s in this data example.

The estimated model 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.

```
> #Simulate Some Data
> 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
> head(simdata2)
```

```
            X Female Y
```

```
1  0.5879062      1 1
2 -0.9276757      1 0
3 -0.7771985      1 1
4  1.1656545      0 1
5 -0.4162322      0 0
6 -0.4445578      0 1
```

```
> # Model:
> myModel <- glm(Y ~ X + Female, data = simdata2, family = "binomial")
```

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 $Female = 1$.
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.
3. We then make 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.

```
> # Generate predictions
> nSims <- 1000
> someScenarios <- expand.grid(1,  # Intercept
+               seq(min(simdata2$X), max(simdata2$X), len = 100),
+               # A sequence covering the range of X values
+               c(0, 1))  # The minimum and maximum of a binary variable
>
> simDraws <- mvrnorm(nSims, coef(myModel), vcov(myModel))
> simYhats <- plogis(simDraws %*% t(someScenarios))
>             #t():matrix transpose function
> dim(simYhats)  #Simulated predictions
```

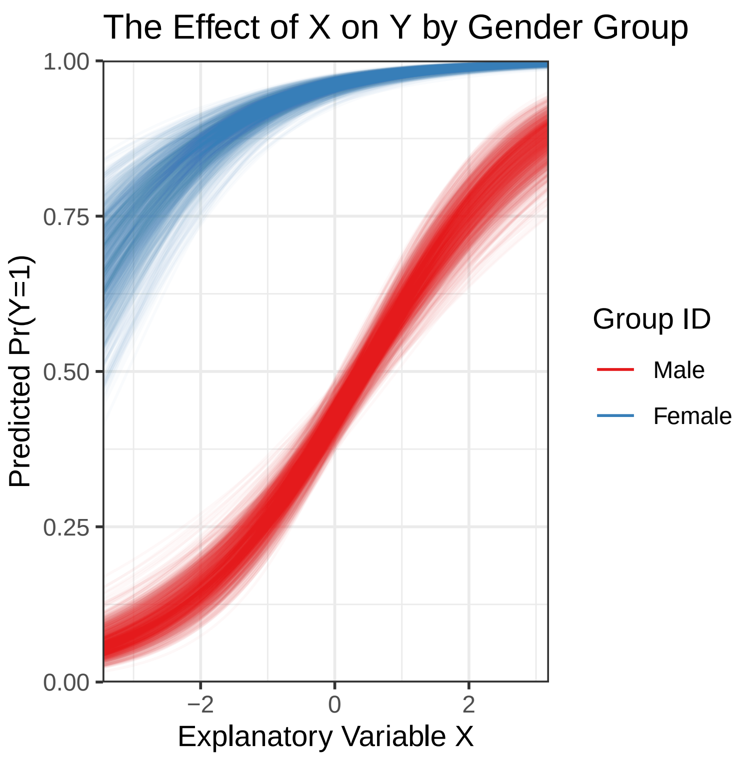
```
[1] 1000  200
```

```
> # 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))
> head(longFrame)
```

```
  Var1      Var2 Var3 variable      value
1    1 -3.456173    0       X1 0.04313749
2    1 -3.389059    0       X1 0.04540329
3    1 -3.321945    0       X1 0.04778216
4    1 -3.254832    0       X1 0.05027910
```

```
5    1 -3.187718    0        X1 0.05289928
6    1 -3.120604    0        X1 0.05564800
```

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


The Effect of X on Y by Gender Group
Predicted Pr(Y=1)
Explanatory Variable X
Group ID
Male
Female