|  |
| --- |
| > # Exercise 1  > # Set a seed  > set.seed(008)  > # create X1 which is a vector of 10,000 draws from a uniform distribution with range 1:3.  > X1 <- as.vector(runif(10000, min = 1, max = 3))  > # create X2 which is a vector of 10,000 draws from a gamma distribution with shape 3 and scale 2  > X2 <- as.vector(rgamma(10000, shape = 3, scale = 2))  > # create X3 which is a vector of 10,000 draws from a binomial distribution with probability 0.3  > X3 <- as.vector(rbinom(10000, size = 1, prob = 0.3))  > # create eps which is a vector of 10,000 draws from a normal distribution with mean 2 and sd 1  > eps <- as.vector(rnorm(10000, mean = 2, sd = 1))  > # create Y and ydum  > Y <- as.vector(0.5 + 1.2\*X1 + -0.9\*X2 + 0.1\*X3 + eps)  > ydum <- as.matrix(as.numeric(Y>mean(Y)))  >  >  > # Exercise 2  > cor(Y, X1)  [1] 0.2021716  > # The output is 0.1452196, which is quite different from 1.2.  > # create a 10000\*1 vector of number 0.5  > X0 <- data.frame(num=1)  > X0 <- as.vector(X0[rep(1:nrow(X0), each=10000),])  > # combine the four vectors  > X <- cbind(X0, X1, X2, X3)  > # calculate the coefficient  > beta1 <- solve(t(X)%\*%X)%\*%t(X)%\*%Y  >  > # Calculate the estimator  > # s2 = (e'e)/n-k(the residual matrix/the degree of freedom), the data can be taken from the output above  > s2 <- as.numeric((t(eps)%\*%eps)/9996)  > # calculate covariance matrix and standard error  > X\_X <- t(X)%\*%X  > X\_X\_2 <- solve(X\_X)  > se\_b <- s2\*X\_X\_2  > se\_b <- sqrt(se\_b)  > # We only need the data on the diagonal  > se\_\_b <- rbind(se\_b[1,1], se\_b[2,2], se\_b[3,3], se\_b[4,4])  >  > # Bootstrap rep = 49  > # combine Y and X  > XY <- cbind(Y,1,X1,X2,X3)  > # create a 49\*4 matrix  > X\_boot <- matrix(c(0,0,0,0),nrow = 49,ncol = 4)  > # use for loop to resample 49 times, put the data into x\_boot  > for(y in 1:49){  + i <- sample(1:10000, size = 10000, replace = TRUE)  + Y <- as.matrix(XY[i,1])  + X1 <- as.vector(XY[i,3])  + X2 <- as.vector(XY[i,4])  + X3 <- as.vector(XY[i,5])  + X4 <- as.matrix(X1,X2,X3)  + X\_boot[y,] <- solve(t(X4)%\*%X4)%\*%t(X4)%\*%Y  + }  > sd\_boot <- as.vector(sd(X\_boot))  >  >  > # Bootstrap rep = 499  > # repeat the process above  > X\_boot2 <- matrix(c(0,0,0,0),nrow = 499,ncol = 4)  > for(y in 1:499){  + i <- sample(1:10000, size = 10000, replace = TRUE)  + Y <- as.vector(XY[i,1])  + X1 <- as.vector(XY[i,3])  + X2 <- as.vector(XY[i,4])  + X3 <- as.vector(XY[i,5])  + X4 <- as.matrix(X1,X2,X3)  + X\_boot2[y,] <- solve(t(X4)%\*%X4)%\*%t(X4)%\*%Y  + }  > sd\_boot2 <- as.vector(sd(X\_boot2))  >  >  > # exercise 3  > # get the log likelihood function  > est\_4 <- function(beta,x,y){  + y <- sum(ydum\*log(pnorm(X%\*%beta))) + sum((1-ydum)\*log(1-pnorm(X%\*%beta)))  + return(-y)  + }  >  >  > # set an eps  > d <- 0.00000001  > # set an initial b  > b <- c(3, 1, -1, 0.05)  > # generate the matrix of b+eps  > bn <- as.matrix(cbind(b,b,b,b))  > bn2 <- diag(d, 4)  > bn3 <- bn + bn2  > # calculate the derivative of log likelihood function and determine the direction  > p1 <- (est\_4(bn3[,1])-est\_4(bn[,1]))/d  > p2 <- (est\_4(bn3[,2])-est\_4(bn[,2]))/d  > p3 <- (est\_4(bn3[,3])-est\_4(bn[,3]))/d  > p4 <- (est\_4(bn3[,4])-est\_4(bn[,4]))/d  > p <- as.vector(c(p1,p2,p3,p4))  > # give an initial change rate  > diff <- 1  > while(diff > 0.0001){  + # generate a new b based on ak and dk, set initial ak = 0.000001  + b0 <- matrix(b, ncol = 1)  + b <- matrix(b + 0.000001\*p, ncol = 1)  + # calculate the change rate of b  + diff <- est\_4(b)-est\_4(b0)  + }  >  > # print b  > b  [,1]  [1,] 2.93733490  [2,] 0.86951812  [3,] -1.35844067  [4,] 0.03191312  > # [,1]  > # [1,] 2.93733490  > # [2,] 0.86951812  > # [3,] -1.35844067  > # [4,] 0.03191312  > # The estimated coefficient of X2 is more different from that in true parameters, compared with the others.  >  >  >  > # exercise 4  > # write the log likelihood function of logit model  > logit\_log <- function(beta, x, y){  + y <- sum(ydum\*log(exp(X%\*%beta)/(1+exp(X%\*%beta)))) + sum((1-ydum)\*log(1-exp(X%\*%beta)/(1+exp(X%\*%beta))))  + return(-y)  + }  > # use optimize function to get the MLE  > xmin\_log <- optim(c(0,0,0,0), logit\_log, x = X, y = ydum)$par  > # use glm function to check the result, the two results are same  > ydumX <- as.data.frame(cbind(ydum, X))  > logit <- glm(ydum ~ 0 + X, family=binomial(link="logit"), data=ydumX)  > summary(logit)  Call:  glm(formula = ydum ~ 0 + X, family = binomial(link = "logit"),  data = ydumX)  Deviance Residuals:  Min 1Q Median 3Q Max  -3.4351 -0.1430 0.0394 0.2609 3.2958  Coefficients:  Estimate Std. Error z value Pr(>|z|)  XX0 5.30949 0.18634 28.494 <2e-16 \*\*\*  XX1 2.22403 0.08216 27.071 <2e-16 \*\*\*  XX2 -1.61703 0.03672 -44.040 <2e-16 \*\*\*  XX3 0.05262 0.08566 0.614 0.539  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  (Dispersion parameter for binomial family taken to be 1)  Null deviance: 13862.9 on 10000 degrees of freedom  Residual deviance: 4315.5 on 9996 degrees of freedom  AIC: 4323.5  Number of Fisher Scoring iterations: 7  > # glm(formula = ydum ~ 0 + X, family = binomial(link = "logit"),  > # data = ydumX)  > #  > # Deviance Residuals:  > # Min 1Q Median 3Q Max  > # -3.4351 -0.1430 0.0394 0.2609 3.2958  > #  > # Coefficients:  > # Estimate Std. Error z value Pr(>|z|)  > # XX0 5.30949 0.18634 28.494 <2e-16 \*\*\*  > # XX1 2.22403 0.08216 27.071 <2e-16 \*\*\*  > # XX2 -1.61703 0.03672 -44.040 <2e-16 \*\*\*  > # XX3 0.05262 0.08566 0.614 0.539  > # ---  > # Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  > #  > # (Dispersion parameter for binomial family taken to be 1)  > #  > # Null deviance: 13862.9 on 10000 degrees of freedom  > # Residual deviance: 4315.5 on 9996 degrees of freedom  > # AIC: 4323.5  > #  > # Number of Fisher Scoring iterations: 7  >  > # interpretation  > # 2.22403 is positive, which means that the probability of success will increase when x1 = 1  > # -1.61703 is negative, which means that the probability of success will decrease when x2 = 1  > # 0.05262 is positive, which means that the probability of success will increase when x3 = 1  > # "\*\*\*" in the summarty means that the estimation of X1 and X2 are signigicant, while that of X3 is not quite significant.  >  > # repeat the process above  > xmin\_pro <- optim(c(0,0,0,0), est\_4, x = X, y = ydum)$par  > probit <- glm(ydum ~ 0 + X, family=binomial(link="probit"), data=ydumX)  > summary(probit)  Call:  glm(formula = ydum ~ 0 + X, family = binomial(link = "probit"),  data = ydumX)  Deviance Residuals:  Min 1Q Median 3Q Max  -3.8746 -0.1081 0.0087 0.2525 3.6264  Coefficients:  Estimate Std. Error z value Pr(>|z|)  XX0 2.92587 0.09857 29.683 <2e-16 \*\*\*  XX1 1.23282 0.04383 28.129 <2e-16 \*\*\*  XX2 -0.89277 0.01818 -49.118 <2e-16 \*\*\*  XX3 0.02977 0.04750 0.627 0.531  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  (Dispersion parameter for binomial family taken to be 1)  Null deviance: 13862.9 on 10000 degrees of freedom  Residual deviance: 4314.1 on 9996 degrees of freedom  AIC: 4322.1  Number of Fisher Scoring iterations: 7  > # glm(formula = ydum ~ 0 + X, family = binomial(link = "probit"),  > # data = ydumX)  > #  > # Deviance Residuals:  > # Min 1Q Median 3Q Max  > # -3.8746 -0.1081 0.0087 0.2525 3.6264  > #  > # Coefficients:  > # Estimate Std. Error z value Pr(>|z|)  > # XX0 2.92587 0.09857 29.683 <2e-16 \*\*\*  > # XX1 1.23282 0.04383 28.129 <2e-16 \*\*\*  > # XX2 -0.89277 0.01818 -49.118 <2e-16 \*\*\*  > # XX3 0.02977 0.04750 0.627 0.531  > # ---  > # Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  > #  > # (Dispersion parameter for binomial family taken to be 1)  > #  > # Null deviance: 13862.9 on 10000 degrees of freedom  > # Residual deviance: 4314.1 on 9996 degrees of freedom  > # AIC: 4322.1  > #  > # Number of Fisher Scoring iterations: 7  >  > # interpretation  > # 1.23282 is positive, which means that the probability of success will increase when x1 = 1  > # -0.89277 is negative, which means that the probability of success will decrease when x2 = 1  > # 0.02977 is positive, which means that the probability of success will increase when x3 = 1  > # "\*\*\*" in the summarty means that the estimation of X1 and X2 are signigicant, while that of X3 is not quite significant.  > # Logit and probit models yield almost the same result.  >  >  > linear\_function <- function(beta, x, y){  + y <- t(ydum-X%\*%beta)%\*%(ydum-X%\*%beta)  + return(y)  + }  > xmin\_linear <- optim(c(0,0,0,0), linear\_function, y = ydum, x = X)$par  > linear\_probability <- lm(ydum~0 + X, data=ydumX)  > summary(linear\_probability)  Call:  lm(formula = ydum ~ 0 + X, data = ydumX)  Residuals:  Min 1Q Median 3Q Max  -0.94585 -0.26742 0.05788 0.24721 2.02094  Coefficients:  Estimate Std. Error t value Pr(>|t|)  XX0 0.8868259 0.0133142 66.608 <2e-16 \*\*\*  XX1 0.1501367 0.0056936 26.370 <2e-16 \*\*\*  XX2 -0.1038651 0.0009539 -108.879 <2e-16 \*\*\*  XX3 0.0059472 0.0072690 0.818 0.413  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  Residual standard error: 0.3306 on 9996 degrees of freedom  Multiple R-squared: 0.8064, Adjusted R-squared: 0.8063  F-statistic: 1.041e+04 on 4 and 9996 DF, p-value: < 2.2e-16  > # lm(formula = ydum ~ 0 + X, data = ydumX)  > #  > # Residuals:  > # Min 1Q Median 3Q Max  > # -0.94585 -0.26742 0.05788 0.24721 2.02094  > #  > # Coefficients:  > # Estimate Std. Error t value Pr(>|t|)  > # XX0 0.8868259 0.0133142 66.608 <2e-16 \*\*\*  > # XX1 0.1501367 0.0056936 26.370 <2e-16 \*\*\*  > # XX2 -0.1038651 0.0009539 -108.879 <2e-16 \*\*\*  > # XX3 0.0059472 0.0072690 0.818 0.413  > # ---  > # Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  > #  > # Residual standard error: 0.3306 on 9996 degrees of freedom  > # Multiple R-squared: 0.8064, Adjusted R-squared: 0.8063  > # F-statistic: 1.041e+04 on 4 and 9996 DF, p-value: < 2.2e-16  >  > # interpretation  > # Since X1 is continuous, 0.1501 is the change in the probability of success given one unit increase in x1.  > # Since X2 is continuous, 0.1501 is the change in the probability of success given one unit increase in x2.  > # Since X3 is discrete, 0.0059 is the difference in the probability of success when x3 = 1 and x3 = 0, holding other xj fixed. It seems that X3 almost doesn't impact the probability.  > # "\*\*\*" in the summarty means that the estimation of X1 and X2 are signigicant, while that of X3 is not quite significant.  >  > # exercise 5  >  > # write the derivative of function of logit  > est\_5 <- function(x){  + y <- exp(x)/(1+exp(x))^2  + return(y)  + }  >  >  > est\_6 <- function(beta){  + # calculate the mean of the X\*beta and f'(mean(XB))  + y1 <- X%\*%beta  + y2 <- mean(y1)  + # calculate the marginal effect  + y3 <- est\_5(y2)\*beta  + y <- as.matrix(y3)  + return(y)  + }  >  > # run the result of Q4  > beta3 <- matrix(c(5.30949, 2.22403, -1.61703, 0.05262), ncol = 1)  > me1 <- est\_6(beta3)  >  >  > # repeat the process above, but in this case, we calculate the detivative theoretically  > est\_7 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + p1 <- dnorm(y2)  + y <- p1\*beta  + y <- as.matrix(y)  + return(y)  + }  >  > beta4 <- matrix(c(2.86589, 1.22988, -0.88943, 0.09990), ncol = 1)  > me2 <- est\_7(beta4)  >  > # calculate the standard deviations using the delta method  > # generate the Jacobian matrix, calculate each partial derivative  > est\_8 <- function(beta){  + # calculate the mean of the X\*beta and f'(mean(XB))  + y1 <- X%\*%beta  + y2 <- mean(y1)  + # calculate the marginal effect  + y3 <- est\_5(y2)\*beta[1,]  + return(y3)  + }  >  > # write the similar functions to calculate the marginal effects respective to beta1, 2, 3 and 4  > # Let's call them ME1\_l, ME2\_l, ME3\_l, ME4\_l  > est\_9 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + y3 <- est\_5(y2)\*beta[2,]  + return(y3)  + }  >  > est\_10 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + y3 <- est\_5(y2)\*beta[3,]  + return(y3)  + }  >  > est\_11 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + y3 <- est\_5(y2)\*beta[4,]  + return(y3)  + }  >  > # calculate the partial derivative of ME1\_l repective to beta1  > beta5 <- matrix(c(5.30950, 2.22403, -1.61703, 0.05262), ncol = 1)  > logit\_11 <- (est\_8(beta3)-est\_8(beta5))/0.00001  > # calculate the partial derivative of ME1\_l repective to beta2  > beta6 <- matrix(c(5.30949, 2.22404, -1.61703, 0.05262), ncol = 1)  > logit\_12 <- (est\_8(beta3)-est\_8(beta6))/0.00001  > # calculate the partial derivative of ME1\_l repective to beta3  > beta7 <- matrix(c(5.30949, 2.22403, -1.61702, 0.05262), ncol = 1)  > logit\_13 <- (est\_8(beta3)-est\_8(beta7))/0.00001  > # calculate the partial derivative of ME1\_l repective to beta4  > beta8 <- matrix(c(5.30949, 2.22403, -1.61703, 0.05263), ncol = 1)  > logit\_14 <- (est\_8(beta3)-est\_8(beta8))/0.00001  > # generate the first row of the Jacobian matrix of logit model  > logit\_1 <- t(as.matrix(c(logit\_11, logit\_12, logit\_13, logit\_14)))  >  > # repeat the process above to generate the second line of this matrix  > logit\_21 <- (est\_9(beta3)-est\_9(beta5))/0.00001  > logit\_22 <- (est\_9(beta3)-est\_9(beta6))/0.00001  > logit\_23 <- (est\_9(beta3)-est\_9(beta7))/0.00001  > logit\_24 <- (est\_9(beta3)-est\_9(beta8))/0.00001  > logit\_2 <- t(as.matrix(c(logit\_21, logit\_22, logit\_23, logit\_24)))  >  > # repeat the process above to generate the third line of this matrix  > logit\_31 <- (est\_10(beta3)-est\_10(beta5))/0.00001  > logit\_32 <- (est\_10(beta3)-est\_10(beta6))/0.00001  > logit\_33 <- (est\_10(beta3)-est\_10(beta7))/0.00001  > logit\_34 <- (est\_10(beta3)-est\_10(beta8))/0.00001  > logit\_3 <- t(as.matrix(c(logit\_31, logit\_32, logit\_33, logit\_34)))  >  > # repeat the process above to generate the fourth line of this matrix  > logit\_41 <- (est\_11(beta3)-est\_11(beta5))/0.00001  > logit\_42 <- (est\_11(beta3)-est\_11(beta6))/0.00001  > logit\_43 <- (est\_11(beta3)-est\_11(beta7))/0.00001  > logit\_44 <- (est\_11(beta3)-est\_11(beta8))/0.00001  > logit\_4 <- t(as.matrix(c(logit\_41, logit\_42, logit\_43, logit\_44)))  >  > # generate the whole Jacobian matrix  > logit\_j <- as.matrix(rbind(logit\_1, logit\_2, logit\_3, logit\_4))  >  > # calculate the sd of logit model: the sd of the diagonal of J(T)\*vcovmatrix\*J  > sd\_delta\_logit <- solve(logit\_j)%\*%vcov(logit)%\*%logit\_j  > sd\_delta\_logit <- sd(rbind(sd\_delta\_logit[1,1], sd\_delta\_logit[2,2], sd\_delta\_logit[3,3], sd\_delta\_logit[4,4]))  >  >  > # write the similar functions to calculate the marginal effects respective to beta1, 2, 3 and 4  > # Let's call them ME1\_p, ME2\_p, ME3\_p, ME4\_p  > est\_12 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + p1 <- dnorm(y2)  + y <- p1\*beta[1,]  + return(y)  + }  >  > est\_13 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + p1 <- dnorm(y2)  + y <- p1\*beta[2,]  + return(y)  + }  >  > est\_14 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + p1 <- dnorm(y2)  + y <- p1\*beta[3,]  + return(y)  + }  >  > est\_15 <- function(beta){  + y1 <- X%\*%beta  + y2 <- mean(y1)  + p1 <- dnorm(y2)  + y <- p1\*beta[4,]  + return(y)  + }  >  > # calculate the partial derivative of ME1\_p repective to beta1  > beta9 <- matrix(c(2.86590, 1.22988, -0.88943, 0.09990), ncol = 1)  > probit\_11 <- (est\_12(beta4)-est\_12(beta9))/0.00001  > # calculate the partial derivative of ME1\_p repective to beta2  > beta10 <- matrix(c(2.86589, 1.22989, -0.88943, 0.09990), ncol = 1)  > probit\_12 <- (est\_12(beta4)-est\_12(beta10))/0.00001  > # calculate the partial derivative of ME1\_p repective to beta3  > beta11 <- matrix(c(2.86589, 1.22988, -0.88942, 0.09990), ncol = 1)  > probit\_13 <- (est\_12(beta4)-est\_12(beta11))/0.00001  > # calculate the partial derivative of ME1\_p repective to beta4  > beta12 <- matrix(c(2.86589, 1.22988, -0.88943, 0.09991), ncol = 1)  > probit\_14 <- (est\_12(beta4)-est\_12(beta12))/0.00001  > # generate the first row of the Jacobian matrix of probit model  > probit\_1 <- t(as.matrix(c(probit\_11, probit\_12, probit\_13, probit\_14)))  >  > # repeat the process above to generate the second line of this matrix  > probit\_21 <- (est\_13(beta4)-est\_13(beta9))/0.00001  > probit\_22 <- (est\_13(beta4)-est\_13(beta10))/0.00001  > probit\_23 <- (est\_13(beta4)-est\_13(beta11))/0.00001  > probit\_24 <- (est\_13(beta4)-est\_13(beta12))/0.00001  > probit\_2 <- t(as.matrix(c(probit\_21, probit\_22, probit\_23, probit\_24)))  >  > # repeat the process above to generate the third line of this matrix  > probit\_31 <- (est\_14(beta4)-est\_14(beta9))/0.00001  > probit\_32 <- (est\_14(beta4)-est\_14(beta10))/0.00001  > probit\_33 <- (est\_14(beta4)-est\_14(beta11))/0.00001  > probit\_34 <- (est\_14(beta4)-est\_14(beta12))/0.00001  > probit\_3 <- t(as.matrix(c(probit\_31, probit\_32, probit\_33, probit\_34)))  >  > # repeat the process above to generate the fourth line of this matrix  > probit\_41 <- (est\_15(beta4)-est\_15(beta9))/0.00001  > probit\_42 <- (est\_15(beta4)-est\_15(beta10))/0.00001  > probit\_43 <- (est\_15(beta4)-est\_15(beta11))/0.00001  > probit\_44 <- (est\_15(beta4)-est\_15(beta12))/0.00001  > probit\_4 <- t(as.matrix(c(probit\_41, probit\_42, probit\_43, probit\_44)))  >  > # generate the whole Jacobian matrix  > probit\_j <- as.matrix(rbind(probit\_1, probit\_2, probit\_3, probit\_4))  >  > # calculate the sd of probit model: the sd of the diagonal of J(T)\*vcovmatrix\*J  > sd\_delta\_probit <- solve(probit\_j)%\*%vcov(probit)%\*%probit\_j  > sd\_delta\_probit <- sd(rbind(sd\_delta\_probit[1,1], sd\_delta\_probit[2,2], sd\_delta\_probit[3,3], sd\_delta\_probit[4,4]))  >  >  > # calculate the standard deviations using bootstrap  > # set an empty matrix to turn in data  > me\_bootlogit <- matrix(c(0,0,0,0),nrow = 450, ncol = 4)  > for(r in 1:450){  + # use bootstrap to generate data for 10000 times  + i <- sample(1:10000, size = 10000, replace = TRUE)  + Y <- as.matrix(ydum[i,])  + X0 <- as.vector(XY[i,2])  + X1 <- as.vector(XY[i,3])  + X2 <- as.vector(XY[i,4])  + X3 <- as.vector(XY[i,5])  + X4 <- as.data.frame(cbind(Y,X0,X1,X2,X3))  + # calculate the coefficient of logit model, then calculate the marginal effect  + betaend <- as.matrix(c(glm(V1 ~ 0 + X, family=binomial(link="logit"), data=X4)$coef))  + me\_bootstraplogit <- est\_6(betaend)  + me\_bootstraplogit <- t(me\_bootstraplogit)  + # put the data into the empty matrix  + me\_bootlogit[r,] <- me\_bootstraplogit  + }  > # calculate sd  > sd\_boot\_logit <- as.vector(sd(me\_bootlogit))  >  >  > # repeat the process above for probit model  > me\_bootprobit <- matrix(c(0,0,0,0),nrow = 450, ncol = 4)  > for(r in 1:450){  + i <- sample(1:10000, size = 10000, replace = TRUE)  + Y <- as.matrix(ydum[i,])  + X0 <- as.vector(XY[i,2])  + X1 <- as.vector(XY[i,3])  + X2 <- as.vector(XY[i,4])  + X3 <- as.vector(XY[i,5])  + X4 <- as.data.frame(cbind(Y,X0,X1,X2,X3))  + betaend2 <- as.matrix(c(glm(V1 ~ 0 + X, family=binomial(link="probit"), data=X4)$coef))  + me\_bootstrapprobit <- est\_7(betaend2)  + me\_bootstrapprobit <- t(me\_bootstrapprobit)  + me\_bootprobit[r,] <- me\_bootstrapprobit  + }  > sd\_boot\_probit <- as.vector(sd(me\_bootprobit)) |
|  |
| |  | | --- | |  | |