Chapter 17: Sample Selection Correction

Susumu Shikano

Last compiled at 18. Juli 2022

Simple Regression under the GM-assumptions

We generate 5000 datasets with n=500 under the GM-assumptions. The number of independent variables is 1. The true regression line has the intercept of 0.4 and the slope of 0.1. The variance of the error is 0.05. The independent variables are generated with the mean 1, variances 0.5 and covariance 0.5.

```
GM.samples <- data.generation(sample.size=sample.size,
                             n.sim=num.datasets,
                             n.iv=n.iv,
                             x.mu=x.mu,
                             x.Sigma=x.Sigma,
                             para=c(true.intercept,true.slope),
                             err.dist = "normal",
                             err.disp = true.err.var,
                             instv = TRUE,
                             instv.num = 1,
                             instv.mu.cov = c(2, 1.5, 0.5))
```

```
## [1] "Covariance with the last independent variable is not zero."
```

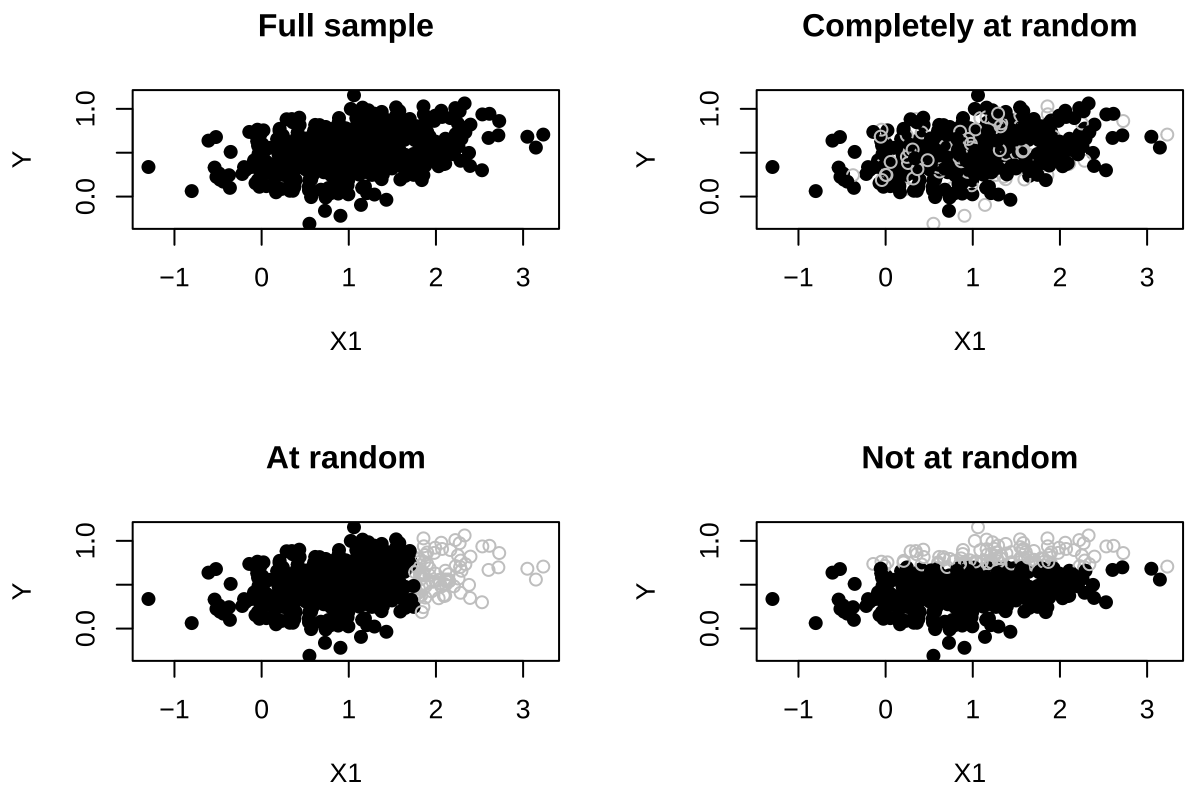
Now, we estimate the simple regression model corresponding to the data generating process however with different samples:

- Full sample (n=500)
- Randomly selected 400 observations (Missing completely at random)
- Only those observations with $x_1 < 1.75$ (Missing at random)
- Only those observations with $y < 0.7$ (Missing not at random)

The figure below displays the above selection using the first sample:

```
n.selected <- 400

all.coef <- array(NA,dim=c(num.datasets,3,6))
all.coef.se <- array(NA,dim=c(num.datasets,2,6))

par(mfrow=c(2,2))

for (i.set in 1:4 ){
for (i in 1:num.datasets){
    this.data <- GM.samples$generated.data[[i]]

    if (i.set==1){
      if (i ==1){
        plot(this.data$y ~ this.data$X1, pch=19,
             main="Full sample",
          xlab="X1",ylab="Y")
  }
}

# sample selection
## completely at random
if (i.set ==2 ){
  selected <- sample(1:nrow(this.data),n.selected)

  if (i ==1){
    this.col <- rep("grey",nrow(this.data))
    this.col[selected]  <- "black"
    this.pch <- rep(1,nrow(this.data))
    this.pch[selected]  <- 19

    plot(this.data$y ~ this.data$X1, pch=this.pch,
         main="Completely at random",
         xlab="X1",ylab="Y",
         col=this.col)
  }

  this.data <- this.data[selected,]
}

## at random
if (i.set ==3 ){
  selected <- ifelse(this.data$X1 < 1.75,TRUE,FALSE)

  if (i ==1){
    this.col <- rep("grey",nrow(this.data))
    this.col[selected]  <- "black"
    this.pch <- rep(1,nrow(this.data))
    this.pch[selected]  <- 19

    plot(this.data$y ~ this.data$X1, pch=this.pch,
         main="At random",
         xlab="X1",ylab="Y",
         col=this.col)
  }

  this.data <- this.data[selected,]
}

## not at random
if (i.set ==4 ){
  selected <- ifelse(this.data$y < 0.7,TRUE,FALSE)

  if (i ==1){
    this.col <- rep("grey",nrow(this.data))
    this.col[selected]  <- "black"
    this.pch <- rep(1,nrow(this.data))
    this.pch[selected]  <- 19
```

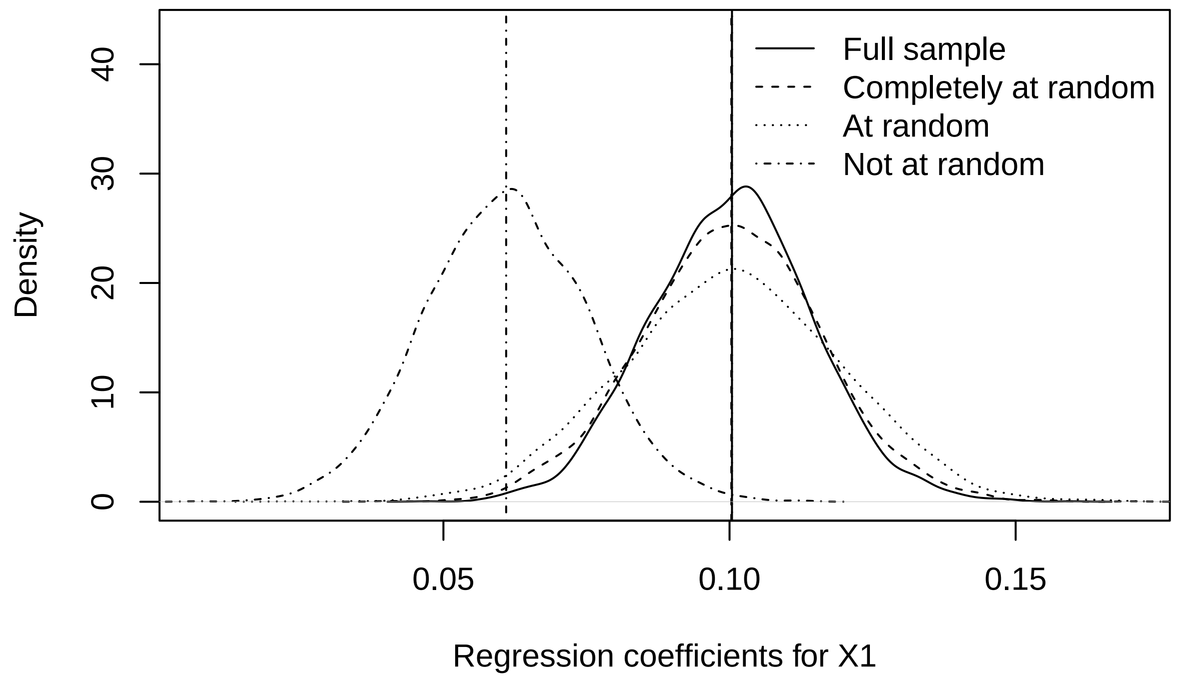
```
        plot(this.data$y ~ this.data$X1, pch=this.pch,
             main="Not at random",
             xlab="X1",ylab="Y",
             col=this.col)
      }

      this.data <- this.data[selected,]

      this.data <- this.data[selected,]
    }


    lm.out <- lm(y ~ X1  ,data=  this.data)

    all.coef[i,1:2,i.set] <- coef(lm.out)
    all.coef[i,3,i.set] <- summary(lm.out)$sigma
    all.coef.se[i,c(1:2),i.set] <- coef(summary(lm.out))[,2]


}
}
```



```
dimnames(all.coef)[[2]] <- c("b0","b1","sigma")
dimnames(all.coef.se)[[2]] <- c("b0","b1")
```

Below, you find the distribution of the point estimates:

```r
x.range <- range(c(all.coef[,"b1",]),na.rm=T)
density.out <- density(all.coef[,"b1",1])
plot(density.out,
     main="",xlab=paste0("Regression coefficients for X1"),
     xlim=x.range,ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",1]))
par(new=T)
plot(density(all.coef[,"b1",2]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=2,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",2]),lty=2)
par(new=T)
plot(density(all.coef[,"b1",3]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=3,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",3]),lty=3)
par(new=T)
plot(density(all.coef[,"b1",4]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=4,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",4]),lty=4)
legend("topright",lty=c(1:4),
       c("Full sample","Completely at random","At random","Not at random"),
       bty="n")
```



Accordingly, all selection processes but the last one (selection based on y) lead to the unbiased estimates.

Incidental truncation

Now, we observe another sample selection process. We model the sample selection mechanism as follows:

$$\hat{s} = 0.25 + 0.5x_1 - 0.3z + \nu$$

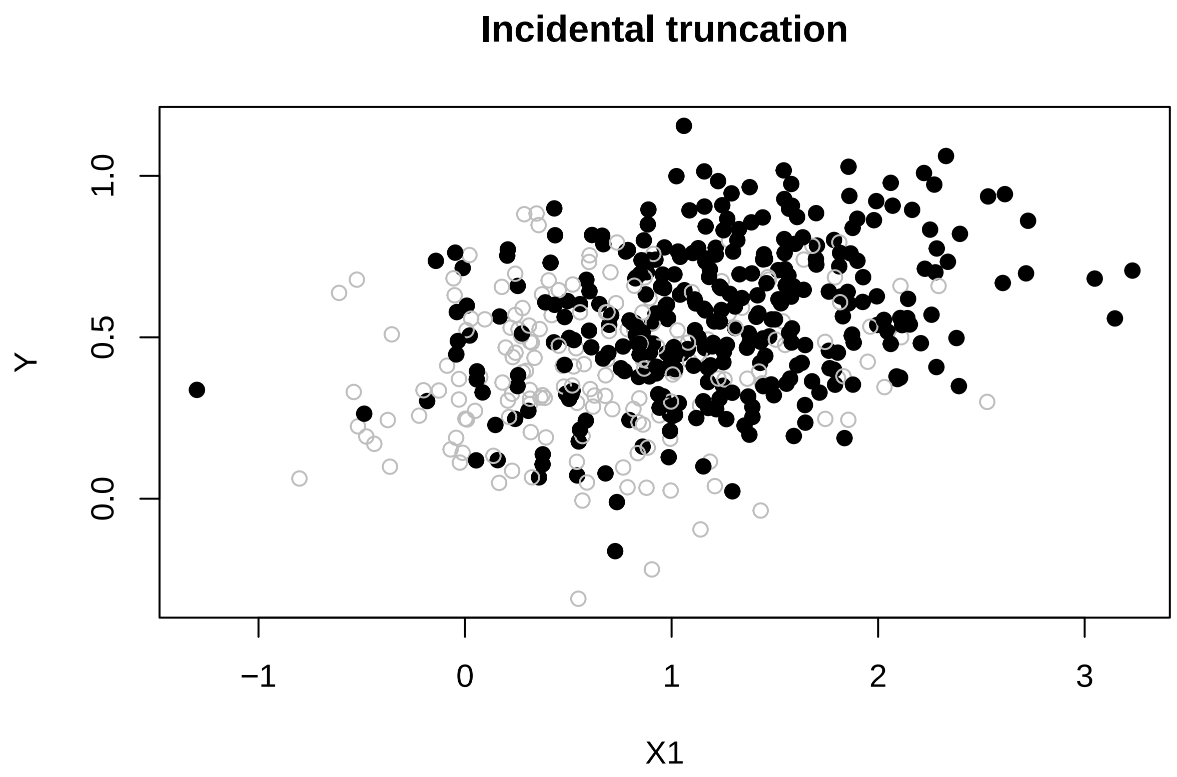
with

$$\nu \sim N(0, 0.04) + u$$

and $s = 1$ (selected) for $\hat{s} \geq 0$ and $s = 1$ (not selected) otherwise.

The figure below displays selected and not selected observations based on a sample data:

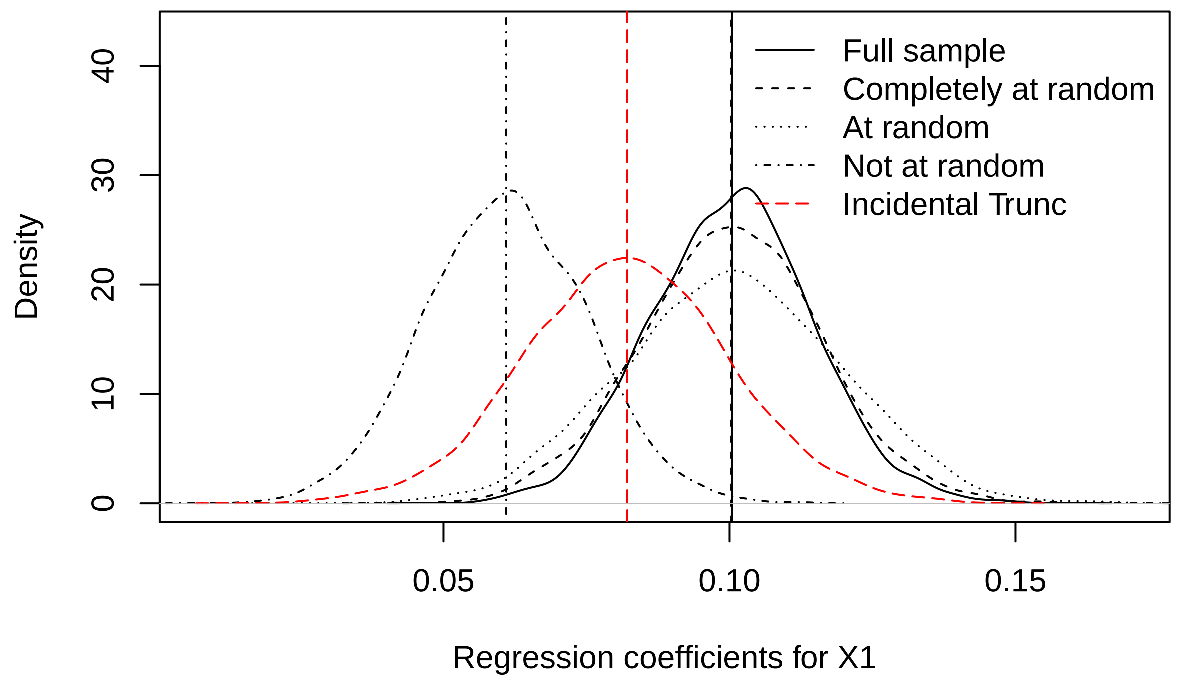
```r
for (i in 1:num.datasets){
    this.data <- this.full.data <- GM.samples$generated.data[[i]]

    # selection model
    nu.err <- rnorm(nrow(this.data),mean=0,sd=0.2) + 0.5*this.data$error

    s.hat <- 0.25 + 0.5*this.data$X1 - 0.3 * this.data$IV + nu.err

    selected <- ifelse(s.hat >= 0,TRUE,FALSE)

    if (i ==1){
    this.col <- rep("grey",nrow(this.data))
    this.col[selected]  <- "black"
    this.pch <- rep(1,nrow(this.data))
    this.pch[selected]  <- 19

    plot(this.data$y ~ this.data$X1, pch=this.pch,
             main="Incidental truncation",
             xlab="X1",ylab="Y",
             col=this.col)
    }

    this.data <- this.data[selected,]


    lm.out <- lm(y ~ X1  ,data=  this.data)

    all.coef[i,1:2,5] <- coef(lm.out)
    all.coef[i,3,5] <- summary(lm.out)$sigma
    all.coef.se[i,c(1:2),5] <- coef(summary(lm.out))[,2]


    # sample selection correction

    prob.out <- glm(selected ~ this.full.data$X1 + this.full.data$IV,
                    family=binomial(link="probit"))

    y.star <-prob.out$linear.predictors
    imr.1 <-  dnorm(-y.star)/(1-pnorm(-y.star))
    imr.1 <- imr.1[selected]

    section.model.out <- lm(y ~ X1 + imr.1  ,data=  this.data)

    all.coef[i,1:2,6] <- coef(section.model.out)[1:2]
    all.coef[i,3,6] <- summary(section.model.out)$sigma
    all.coef.se[i,c(1:2),6] <- coef(summary(section.model.out))[1:2,2]

}
```



Using the selected samples, we estimate the same simple regression model above. The figure below displays the estimate distribution in red:

```
x.range <- range(c(all.coef[,"b1",]),na.rm=T)
density.out <- density(all.coef[,"b1",1])
plot(density.out,
     main="",xlab=paste0("Regression coefficients for X1"),
     xlim=x.range,ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",1]))
par(new=T)
plot(density(all.coef[,"b1",2]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=2,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",2]),lty=2)
par(new=T)
plot(density(all.coef[,"b1",3]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=3,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",3]),lty=3)
par(new=T)
plot(density(all.coef[,"b1",4]),ann=F,xlab="",ylab="",main="",
     axes=F,
     xlim=x.range,lty=4,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",4]),lty=4)
```

```
    par(new=T)
    plot(density(all.coef[,"b1",5]),ann=F,xlab="",ylab="",main="",
         axes=F,col="red",
         xlim=x.range,lty=5,
         ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",5]),lty=5,col="red")
    legend("topright",lty=c(1:5),
           c("Full sample","Completely at random","At random","Not at random",
             "Incidental Trunc"),
           col=c(rep("black",4),"red"),
           bty="n")
```



It is obvious that the estimates are biased. This is because we have the error term ν in the selection process which correlates with u, the error of the main regression model.

If it is known what variables are the part of the selection mechanism (x_1 and z in the above equation), we can utilize the Heckit method, which are presented below.

First, we model the selection by x_1 and z (in the code IV) using the probit model:

```
prob.out <- glm(selected ~ this.full.data$X1 + this.full.data$IV,
                    family=binomial(link="probit"))
summary(prob.out)
```

```
##
## Call:
## glm(formula = selected ~ this.full.data$X1 + this.full.data$IV,
##     family = binomial(link = "probit"))
```

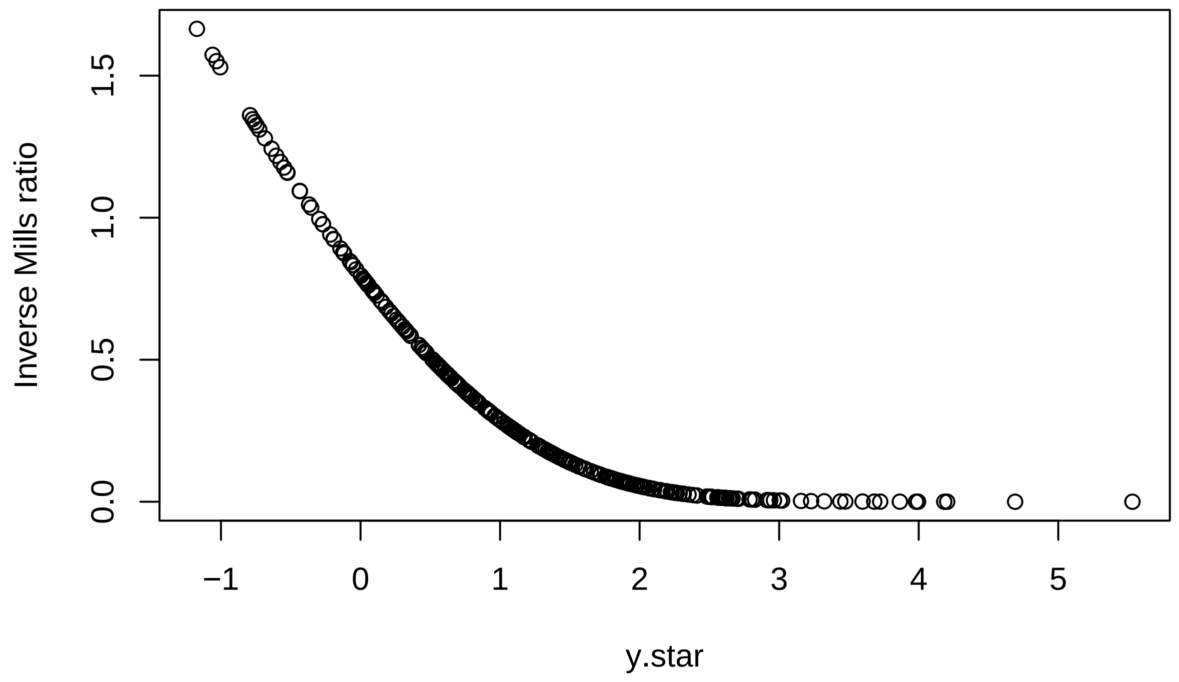
```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1642  -0.5311   0.1793   0.6479   2.0571
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.9947     0.1547   6.431 1.27e-10 ***
## this.full.data$X1   2.1997     0.1854  11.867  < 2e-16 ***
## this.full.data$IV  -1.2813     0.1124 -11.397  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 654.56  on 499  degrees of freedom
## Residual deviance: 385.30  on 497  degrees of freedom
## AIC: 391.3
##
## Number of Fisher Scoring iterations: 6
```

Based on the result, we obtain the linear predictors:

$$y^* = \gamma_0 + \gamma_1 x_1 + \gamma_2 z$$

This predictors are rescaled by using the inverse Mills ratio.

```r
y.star <-prob.out$linear.predictors
imr.1 <-  dnorm(-y.star)/(1-pnorm(-y.star))
imr.1 <- imr.1[selected]

plot(imr.1 ~ y.star[selected],xlab="y.star",ylab="Inverse Mills ratio")
```



Now, we can regress y on x_1 and the inverse Mills ratio to obtain the corrected estimate for β_1. Note that we use only the selected samples in this regression:

```
section.model.out <- lm(y ~ X1 + imr.1  ,data=  this.data)

summary(section.model.out)
```

```
##
## Call:
## lm(formula = y ~ X1 + imr.1, data = this.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67446 -0.15255  0.00292  0.15040  0.56860
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.39323    0.02889  13.612  < 2e-16 ***
## X1           0.10461    0.01793   5.834 1.34e-08 ***
## imr.1        0.11544    0.03581   3.224   0.0014 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2159 on 316 degrees of freedom
## Multiple R-squared:  0.1027, Adjusted R-squared:  0.09706
## F-statistic: 18.09 on 2 and 316 DF,  p-value: 3.643e-08
```

If one compares the result with the naive simple regression model, $\hat{\beta}_1$ is significantly different:

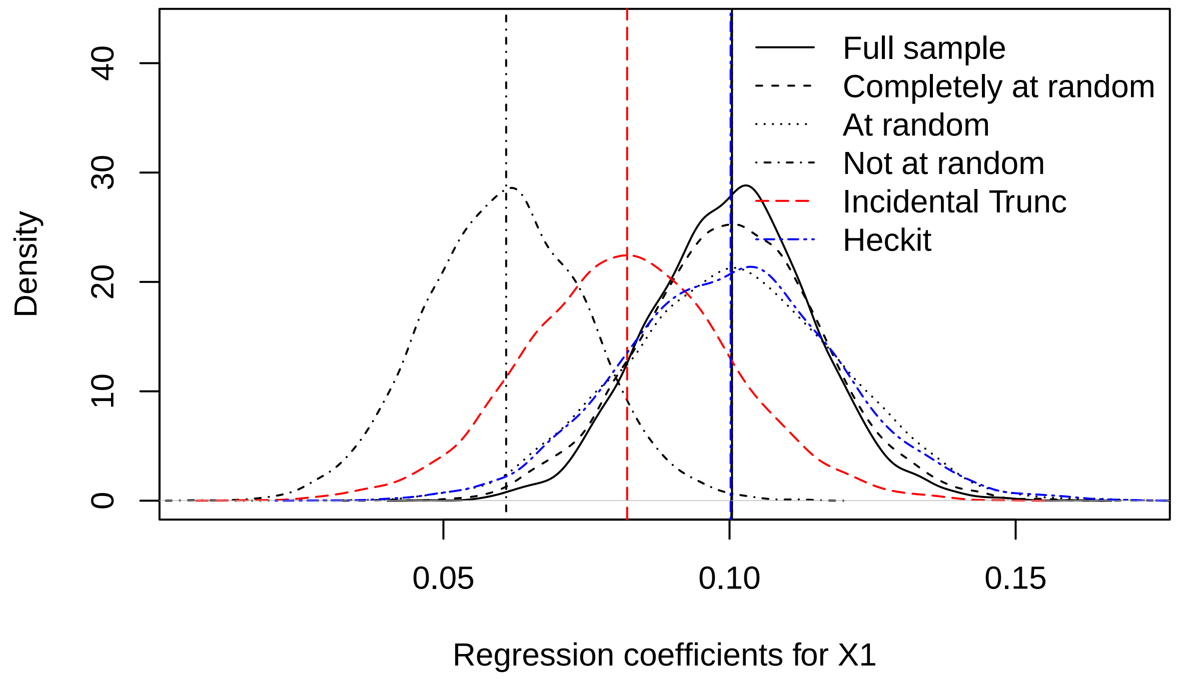
```
summary(lm.out)
```

```
##
## Call:
## lm(formula = y ~ X1, data = this.data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.6201 -0.1560 -0.0044  0.1553  0.5712 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.45226    0.02267  19.947  < 2e-16 ***
## X1           0.08644    0.01727   5.004 9.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2191 on 317 degrees of freedom
## Multiple R-squared:  0.07322,    Adjusted R-squared:  0.0703 
## F-statistic: 25.04 on 1 and 317 DF,  p-value: 9.305e-07
```

The estimates based on the Heckit method are presented in blue in the digure below:

```
    x.range <- range(c(all.coef[,"b1",]),na.rm=T)
    density.out <- density(all.coef[,"b1",1])
    plot(density.out,
         main="",xlab=paste0("Regression coefficients for X1"),
         xlim=x.range,ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",1]))
    par(new=T)
    plot(density(all.coef[,"b1",2]),ann=F,xlab="",ylab="",main="",
         axes=F,
         xlim=x.range,lty=2,
         ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",2]),lty=2)
    par(new=T)
    plot(density(all.coef[,"b1",3]),ann=F,xlab="",ylab="",main="",
         axes=F,
         xlim=x.range,lty=3,
         ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",3]),lty=3)
    par(new=T)
    plot(density(all.coef[,"b1",4]),ann=F,xlab="",ylab="",main="",
         axes=F,
         xlim=x.range,lty=4,
         ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",4]),lty=4)
    par(new=T)
    plot(density(all.coef[,"b1",5]),ann=F,xlab="",ylab="",main="",
         axes=F,col="red",
         xlim=x.range,lty=5,
         ylim=c(0,max(density.out$y)*1.5))
    abline(v=mean(all.coef[,"b1",5]),lty=5,col="red")
    par(new=T)
```

```
plot(density(all.coef[,"b1",6]),ann=F,xlab="",ylab="",main="",
     axes=F,col="blue",
     xlim=x.range,lty=6,
     ylim=c(0,max(density.out$y)*1.5))
abline(v=mean(all.coef[,"b1",6]),lty=6,col="blue")
legend("topright",lty=c(1:6),
       c("Full sample","Completely at random","At random","Not at random",
         "Incidental Trunc","Heckit"),
       col=c(rep("black",4),"red","blue"),
       bty="n")
```



The distribution shows that the Heckit method successfully correct the estimates and made them unbiased.