SemiParametric Estimation Using R

Deng Dongsheng
ID: 13210680046

January 1, 2015

1 Problem

Consider a model

$$y = b_0 + b_1 x_1 + b_2 x_2 + z_1 + z_2 + z_1 z_2 + u \tag{1}$$

where $b_0 = b_1 = b_2 = 1, x_1 \sim \mathcal{N}(0,1), x_2 \sim \mathcal{N}(0,1), u \sim \mathcal{N}(0,1)$, they are independent with each other. z_1, z_2 satisfy that

$$z_1 \sim \sqrt{0.4} z_{11} + \sqrt{0.6} z_{12} \tag{2}$$

$$z_2 \sim \sqrt{0.4} z_{11} + \sqrt{0.6} z_{13} \tag{3}$$

where (z_{11}, z_{12}, z_{13}) are standard normal vector.

1. Generate a random sample of size 1000
2. The regression model is

   $$y = b_1 x_1 + b_2 x_2 + g(z_1, z_2) + u \tag{4}$$

   Question: Using the *Density–Weighted* method to estimate partially linear model. What are your estimates of b_1 and b_2? What are the asymptotic standard error of $\hat{b}_1, \hat{b}_2$?
3. What is the optimal bandwidth? What's your estimate of $g(z_1, z_2 | z_1 = 0.5, z_2 = 1.2)$? Draw a graph of the estimated curve of $g(z_1, z_2 | z_2 = 1)$.
4. Redo (2) and (3) using a sample size of 10000, what's the difference compare to the results of size 1000?

2 Data Generating Process

We need to generate 6 series of number whose size is 1000 from standard normal distribution, in order to make the results reproduciable, we use different seeds to generate them since they are independent with each other.

I define a function to generate the normal distribution samples called "gnorm", use sapply and dplyr package, we generate the **TRUE** data.

```
library(dplyr)
b0 <- 1; b1 <- 1; b2 <- 1
size <- 1000
seed <- c(123,323,432,378,245,765)
gnorm <- function(seed) {
    set.seed(seed)
    return(rnorm(size))
```

```
}
data <- as.data.frame(sapply(seed,gnorm))
colnames(data) <- c("x1","x2","z11","z12","z13","u")
data <- mutate(data,
                    z1 = sqrt(0.4)*z11 + sqrt(0.6)*z12,
                    z2 = sqrt(0.4)*z11 + sqrt(0.6)*z13)
data <- mutate(data,
                    y = b0 + b1*x1 + b2*x2 + z1 + z2 + z1*z2 +u)
```

3 Estimates

Once we get the data, we choose x_1 and x_2 to construct X, and z_1, z_2 to construct Z, and y for Y. This process is just for simplicity of data manipulation.

```
X <- cbind(data$x1,data$x2)
Z <- cbind(data$z1,data$z2)
Y <- data$y
```

At first, we define two functions, named "ker" and "Kh". "ker" is standard normal kernel (18). "Kh" is to compute the value of $K_h(Z_i, Z_j)$ according to (11).

```
ker <- function(v) {
    return(dnorm(v)) # std normal kernel
    # ifelse(abs(v)<=1, 0.5, 0) # uniform kernel
}
Kh <- function(Zi,Zj,h=h) {
    return(prod(sapply((Zi-Zj)/h,ker))/prod(h))
}
```

If we choose the bandwidth h which the rule of thumb indicates, we have $h = (0.3303423, 0.3247018)$, given the bandwidth, we can estimate the linear parametric part. In order to avoid writing to the computer memory all the time, we initial with a data container to store the data generated in the process of estimation.

```
XY <- data.frame(x1h = rep(NA,size),
                 x2h = rep(NA,size),
                  fz = rep(NA,size),
                  yh = rep(NA,size),
                  x1 = data$x1,
                  x2 = data$x2,
                   y = data$y)
```

Here, x1h is the column names for variable $\hat{x}_1$, x2h is the column names of $\hat{x}_2$, and $\hat{X}_i = (\hat{x}_{i1}, \hat{x}_{i2})$. fz is the name of $\hat{f}(Z)$, yh is the column names of $\hat{y}$.

Then we can define a function to estimate $\beta = (b_1, b_2)$, here it's "estimatebeta". For the first part of the function, we have two for loops, using the two for loops, we can calculate $\hat{Y}_i$, $\hat{X}_i$, $\hat{f}(Z_i)$ for $i = 1, 2, \ldots, 1000$.

```
estimatebeta <- function(h, method = "dwe") {
    for (i in 1:size){
        HYi <- 0; FZi <- 0; HXi <- 0
```

```
        for (j in 1:size) {
            Khij <- Kh(Z[i,],Z[j,],h)
            FZi <- Khij + FZi
            HXi <- X[j,]*Khij + HXi
            HYi <- Y[j]*Khij + HYi
        }
        XY[i, 3] <- FZi/size
        XY[i,c(1,2)] <- HXi/(size*FZi)
        XY[i, 4] <- HYi/(size*FZi)
    }
    if (method == "default") {
        XY <- mutate(XY, x1d = (x1 - x1h),
                         x2d = (x2 - x2h),
                          yd = ( y - yh))
        mod <- lm(yd~x1d+x2d-1,XY)
    } else { # Density Weighted Estimator
        XY <- mutate(XY, x1f = (x1 - x1h)*fz,
                         x2f = (x2 - x2h)*fz,
                          yf = ( y - yh)*fz)
        mod <- lm(yf~x1f+x2f-1,XY)
    }
    return(mod$coef)
```

I use different methods to estimate β, we have results illustrated in Table 1. $h = (0.214, 0.284)$ is the optimal bandwidth using density weighted method to minimize the LSCV. It seems that it does not vary too much using different methods to estimate the parameter. Nevertheless, if we use Robinson's method to estimate β when minimizing LSCV, it will result in another "optimal" bandwidth h', the result is much bad, and $\hat{\beta}$ deviate from 1 too much that I didn't recommend to use. In sum, given the right "optimal" bandwidth, either methods is OK. In table 1, the numbers in braces is the standard error of the estimates. uni ker is short for "uniform kernel" and std ker for "standard kernel".

In the following part, I will use **Density Weighted Method** together with standard normal kernel to estimate.

Table 1: Estimates of β with different kernels and methods

Methods		Robinson's Method				Density Weighted Method			
kernel	bandwidth	$h = (0.330, 0.325)$		$h = (0.214, 0.284)$		$h = (0.330, 0.325)$		$h = (0.214, 0.284)$	
beta	TRUE	uni ker	std ker	uni ker	std ker	uni ker	std ker	uni ker	std ker
b_1	1	1.0549	1.05494	1.05491	1.05492	0.93704	0.94917	0.94458	0.94394
	-	(0.0809)	(0.0809)	(0.0809)	(0.0809)	(0.0570)	(0.0573)	(0.0570)	(0.0571)
b_2	1	0.94332	0.94331	0.94332	0.94333	1.03740	1.03909	1.02854	1.04141
	-	(0.0797)	(0.0797)	(0.0797)	(0.0797)	(0.0546)	(0.0551)	(0.0540)	(0.0548)

In previous context, I used the "optimal" bandwidth $h = (0.2144863, 0.2838966)$. In fact, it's the argument which minimize the least squares cross-validation (see (15)). In order to estimates β and $h = (h_1, h_2)$ simultaneously, I define a function "lscv" which calls "estimatebeta" function when minimizing LSCV.

```
lscv <- function(h) {
    LSCV = 0 # initial LSCV
    betahat <- as.matrix(estimatebeta(h, method = "dwe"))
```

```
    for (i in 1:size) {
        Gzi <- 0; Khi <- 0;
        for (j in seq(size)[-i]) {
            Gzi <- Gzi + Kh(Z[i,],Z[j,],h)*(Y[j] - (X[j,] %*% betahat))
            Khi <- Khi + Kh(Z[i,],Z[j,],h)
        }
        gzi <- Gzi/Khi
        LSCV <- (Y[i] - (X[i,] %*% betahat) - gzi)^2 + LSCV
    }
    return(LSCV)
}
```

The function calls function "estimatebeta" to obtain the value of $\hat{\beta}$ given h, and given $\hat{\beta}$, "lscv" function return the value of LSCV, then we can use "optim" function to minimize the value of LSCV around $h = (0.3303423, 0.3247018)$, i.e. the recommended bandwidth from **Rule of Thumb**.

```
h0 <- c(0.3303423, 0.3247018)
opth <- optim(h0,lscv)
```

It takes much time to run the optim function when the size is set to be 1000, the optimal bandwidth is $h = (0.2144863, 0.2838966)$.

When $z_1 = 0.5, z_2 = 1.2$, the actual $g(z_1, z_2|z_1 = 0.5, z_2 = 1.2) = 1 + 0.5 + 1.2 + 0.5 * 1.2 = 3.3$, what's the estimate? I define a function named "cgz" to compute the nonparametric component (13)

```
cgz <- function(z) {
    Gzi <- 0; Khi <- 0;
    for (j in seq(size)) {
        Gzi <- Gzi + Kh(z,Z[j,],h)*(Y[j] - (X[j,] %*% betahat))
        Khi <- Khi + Kh(z,Z[j,],h)
    }
    gzi <- Gzi/Khi
    return(gzi)
}
```

Using the function cgz(z=c(0.5,1.2)), we know $\hat{g}(z_1, z_2|z_1 = 0.5, z_2 = 1.2) = 3.257743$.

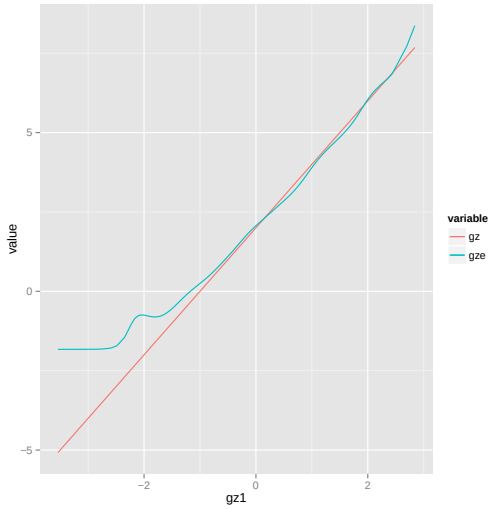
Then we turn to calculate the $g(z_1, z_2|z_2 = 1)$. Since $z_1 = \sqrt{0.4}z_{11} + \sqrt{0.6}z_{12}$, I reproduce the standard normal samples with seed equal 1234 and 4345.

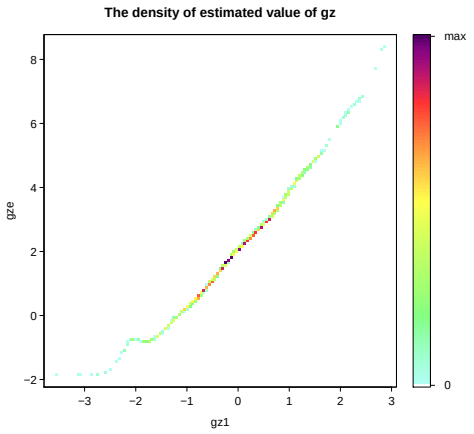
```
sizez <- 1000
set.seed(1234)
gz11 <- rnorm(sizez)
set.seed(4345)
gz12 <- rnorm(sizez)
sdata <- data.frame(gz11=gz11,gz12=gz12)
sdata <- mutate(sdata,
                gz1 = sqrt(0.4)*gz11+sqrt(0.6)*gz12,
                gz2 = 1,
                gz = b0 + gz1 + gz2 + gz1*gz2)
gze <- rep(NA,sizez)
```

With the origin data "sdata"(gz_1) and estimated data container "gze", loop over 1 to sizez to get the estimated $g(z_1, z_2 \mid z_2 = 1)$, we can plot the TRUE data and estimated data in a single figure for good comparison.

```
for (i in seq(sizez)) {
    gze[i] <- cgz(c(sdata$gz1[i],1))
}
sdata <- mutate(sdata, gze = gze)
sdata <- arrange(sdata,gz1)
library(ggplot2)
df <- data.frame(gz1 = sdata$gz1, gz = sdata$gz, gze = sdata$gze)
ggplot(df, aes(gz1, y = value, color = variable)) +
    geom_line(aes(y = gz, col = "gz")) +
    geom_line(aes(y = gze, col = "gze"))
```

The left figure shows the comparison of the true curve of $g(z_1, z_2|z_2 = 1)$ and the estimated curve. In the figure, gz is the true, and gze is the estimated curve. Notice the accuracy of the nonparametric estimates depends on the distance between z_1 and 0 (the expectation of z_1). We know most of the points are centered at 0 (right figure), thus the error is not very large.





4 Larger sample size?

Consider a larger sample size (= 10000), since it may takes few days to run the program, we have bootstraps method to calculate the optimal bandwidth h, in our sample selection, we randomly choose 100 from the population 10000 for 100 times. Every time k, we compute the constant c_{1k} and c_{2k} such that

$$h_k^{\text{opt}} = (c_{1k} Z_{1k}^{\text{sd}} n^{-1/6}, c_{2k} Z_{2k}^{\text{sd}} n^{-1/6}) \tag{5}$$

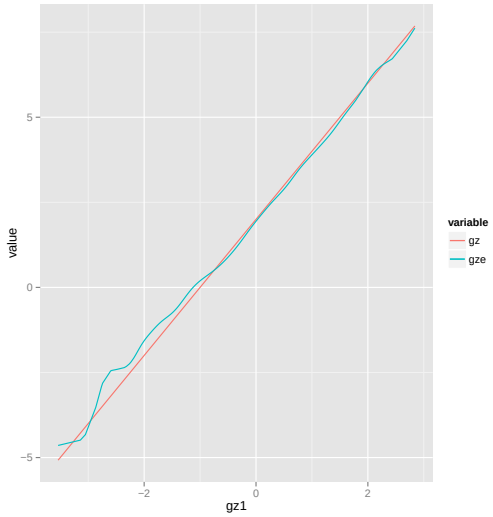
where $n = 100$ is the sample size we randomly chosen, $k(= 1, \ldots, 100)$ is the seclection index. h_k^{opt} is the optimal bandwidth when considering the LSCV in our subsample k whose size is 100.

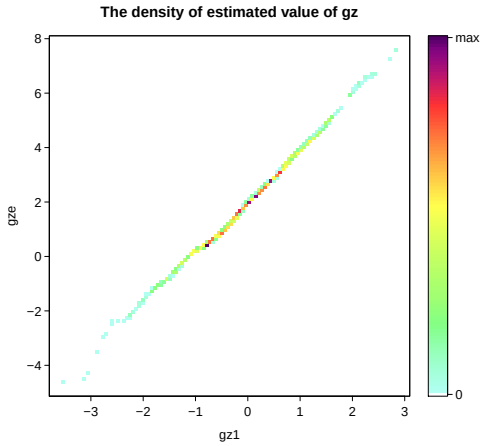
Once we get $c_k = (c_{1k}, c_{2k})$ (the data is stored in "hs.csv"), we use the mean of the c_k ($\bar{c} = (0.8448261, 0.8648953)$) to compute the optimal bandwidth of the full sample ($h = (0.1799968, 0.1845503)$). Given the optimal bandwidth h, we can get $\beta = (b_1, b_2)$ estimated. With Robinson's method and Density Weighted method, we have results presented in table 2. As the table indicates, the standard error of β reduces as the sample size increasing from 1000 to 10000, the estimates converges to 1 more accurately. We can use "cgz" function to compute $\hat{g}(z_1, z_2|z_1 = 0.5, z_2 = 1.2) = 3.145081$. As the previous part shows, we can get the estimated nonparametric

Table 2: Estimation of β when sample size is 10000

estimates	method	Robinson Method	Density Weighted Method
b_1	1	1.0063	0.97582
	-	(0.02568)	(0.01846)
b_2	1	1.00761	0.99749
	-	(0.02555)	(0.01831)

component, the following figure shows the estimated curve $\hat{g}(z_1, z_2|z_2 = 1)$. It turns out to be more accurate as the sample size becomes 10000.





5 Semiparametric Partially Linear Model Theory

5.1 Estimation of Parametric Linear Component

A semiparametric partially linear model is given by

$$Y_i = X_i'\beta + g(Z_i) + u_i, \quad i = 1, \ldots, n, \tag{6}$$

We have feasible estimation of β given by

$$\hat{\beta} = \Big[\sum_{i=1}^{n} \tilde{X}_i \tilde{X}_i'\Big]^{-1} \sum_{i=1}^{n} \tilde{X}_i \tilde{Y}_i. \tag{7}$$

where $\tilde{X}_i = X_i - \hat{X}_i$, and $\tilde{Y}_i = Y_i - \hat{Y}_i$, and

$$\hat{Y}_i \equiv \hat{\mathbf{E}}(Y_i|Z_i) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{j=1}^{n} Y_j K_h(Z_i, Z_j)/\hat{f}(Z_i) \tag{8}$$

$$\hat{X}_i \equiv \hat{\mathbf{E}}(X_i|Z_i) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{j=1}^{n} X_j K_h(Z_i, Z_j)/\hat{f}(Z_i) \tag{9}$$

and

$$\hat{f}(Z_i) = \frac{1}{n}\sum_{j=1}^{n} K_h(Z_i, Z_j) \tag{10}$$

where

$$K_h(Z_i, Z_j) = \prod_{s=1}^{q} \frac{1}{h_s} k(\frac{Z_{is} - Z_{js}}{h_s}). \tag{11}$$

With Density–Weighted method (By Li 1996), we have

$$f(Z_i)\tilde{Y}_i = \left(f(Z_i)\tilde{X}_i\right)'\beta + f(Z_i)u_i \tag{12}$$

then use OLS to estimate β.

5.2 Estimation of the Nonparametric Component

We know $g(Z_i) = \mathbf{E}(Y_i - X_i'\beta|Z_i)$. Therefore, after obtaining a $\sqrt{n}-$ consistent estimator of β, a consistent estimator of $g(z)$ is given by

$$\hat{g}(z) = \frac{\sum_{j=1}^{n}(Y_i - X_j'\hat{\beta})K_h(z, Z_j)}{\sum_{j=1}^{n} K_h(z, Z_j)}. \tag{13}$$

5.3 Optimal bandwidth using LSCV

All the previous theory part is based on the key variable h, we can use the h which the **Rule of Thumb** indicates, i.e.

$$h \approx 1.06\sigma n^{-1/(q+4)} \approx 1.06 X_{sd} n^{-1/(q+4)} \tag{14}$$

More precisely, we use least squares cross-validation (LSCV) to choose $h_1, \ldots, h_q$. That is, one can always choose $h_1, \ldots, h_q$ to minimize

$$\sum_{i=1}^{n} \left[Y_i - X_i'\hat{\beta} - g_{-i}(Z_i, h)\right]^2 \tag{15}$$

where

$$\hat{g}_{-i}(Z_i, h) = \frac{\sum_{j\neq i}(Y_j - X_j'\hat{\beta})K_h(Z_i, Z_j)}{\sum_{j\neq i} K_h(Z_i, Z_j)} \tag{16}$$

We can choose h and estimate β simultaneously since once we get h, we can get the estimates of β.

5.4 Kernel Function

We can use different kernel functions to estimate the pdf.

A uniform kernel function is given by

$$k(z) = \begin{cases} 1/2 & \text{if } |z| \leq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

A standard normal kernel is given by

$$k(v) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}v^2} \tag{18}$$

In my code, one can choose either way to use. By default, I use standard normal kernel, if you want to use uniform kernel, just comment the line 16 in *semi.R*, and uncomment the line 17.