A cautionary tale on instrument vector calibration for the treatment of nonignorable unit nonresponse in surveys

Eric Lesage, David Haziza and Xavier D'Haultfoeuille

July 5, 2017

R code corresponding to the empirical investigation presented in Section 6

Supplementary material

1. Function used to generate the proxy variables x

```
gen_proxy<- function(gamma0, gamma1, gamma2, z, u)
            {
            sigmax=(1 - gamma1^2 - gamma2^2)^(0.5)
            return(gamma0 * rep(1, N) + gamma1 * z + gamma2 * u
            + sigmax * rnorm(n = N, mean=0, sd=1))
            }
```

2. Parameters

```
#1. Population size and number of iterations
set.seed(1934)        # seed used to find exactly the same results
                      # each time the simulation is performed
N       = 1000                    # Population size
K       = 10000           # Number of Monte Carlo replicates

#2. Values of the paramaters gamma_1 and gamma_2 for the proxy variables
Gamma_0     = 0
Gamma_1     = c(0.6, 0.4, 0.2)              # dim = 3
Gamma_2     = c(0.0, 0.1, 0.3, 0.5 )        # dim = 4
```

3. Variables formats

```
# Variables
X   = array(data = 0, dim = c(3, 4, N))             # dim = 3 * 4 * N
y1  = rep(0, N)           # First survey variable
y2  = rep(0, N)           # Second survey variable
one = rep(1, N)         # Vector of 1, dummy variable
R   = rep(0, N)           # Vector of nonresponse indicator
```

```
# True values and estimators
# ty1: true total
ty1     = rep(0, K)
# ty1c: instrumental calibration estimator
ty1c    = array(data = 0, dim = c(3, 4, K))         # dim = 3 * 4 * K
# ty1n: naive estimtor
ty1n    = rep(0, K)
# ty1conv: conventional calibration estimator
ty1conv = array(data = 0, dim = c(3, 4, K))         # dim = 3 * 4 * K

# The same for the variable y2
ty2     = rep(1, K)
ty2c    = array(data = 0, dim = c(3, 4, K))         # dim = 3 * 4 * K
ty2n    = rep(1, K)
ty2conv = array(data = 0, dim = c(3, 4, K))         # dim = 3 * 4 * K
```

Intrument Vector Calibration and Conventional Vector Calibration

```
# We perform K = 10 000 replicates
library(sampling)   # We use the function gencalib() of
                    # the package sampling to get the calibration weights
for (j in 1:K)
  {
  # Generation of the variables U and Z
  z = sqrt(3) * runif(n=N, min = -1 , max = 1)
  u = sqrt(3) * runif(n=N, min = -1 , max = 1)
  # Generation of the variable Y_1k: Linear model
  y1 = 10 + 5 * z + rnorm(n=N, mean=0, sd=2)
  # Generation of the variable Y_2k: Exponential model
  y2 = exp(2.5*z) + rnorm(n=N, mean=0, sd=2)
  # Generation of the variables p and R
  p = 1 / (2 + 0.35 * z)  +  0.1 * u
  R = rbinom( n = N, size = 1, prob = p)
  # Finite population parameters: t_y1  and t_y2
  ty1[j]    = sum(y1)
  ty2[j]    = sum(y2)
  # Estimators
  for (l in 1:3)
    {
    for (c in 1:4)
      {
      # Generation of the variables the 12 X(gamma_1, gamma_2)
      X[l,c,] = gen_proxy(Gamma_0, Gamma_1[l] , Gamma_2[c], z, u)
      # Naive Estimator
      ty1n[j] = N * mean(y1[(R==1)])
      ty2n[j] = N * mean(y2[(R==1)])
      # 12 Instrumental calibration estimators
      total = c(N, sum(X[l,c,]))
      Fcal  = gencalib( cbind(one[(R==1)], X[l,c,(R==1)]),
                        cbind(one[(R==1)], z[(R==1)]),
                        d = one[(R==1)], total, method = "linear")
```

```
      ty1c[l,c,j] = t(Fcal) %*% y1[(R==1)]
      ty2c[l,c,j] = t(Fcal) %*% y2[(R==1)]
      # 12 Conventional calibration estimators
      Fcal2 = calib(cbind(one[(R==1)], X[l,c,(R==1)]),
                    d = one[(R==1)], total, method = "linear")
      ty1conv[l,c,j] = t(Fcal2) %*% y1[(R==1)]
      ty2conv[l,c,j] = t(Fcal2) %*% y2[(R==1)]
      }
    }
  }
```

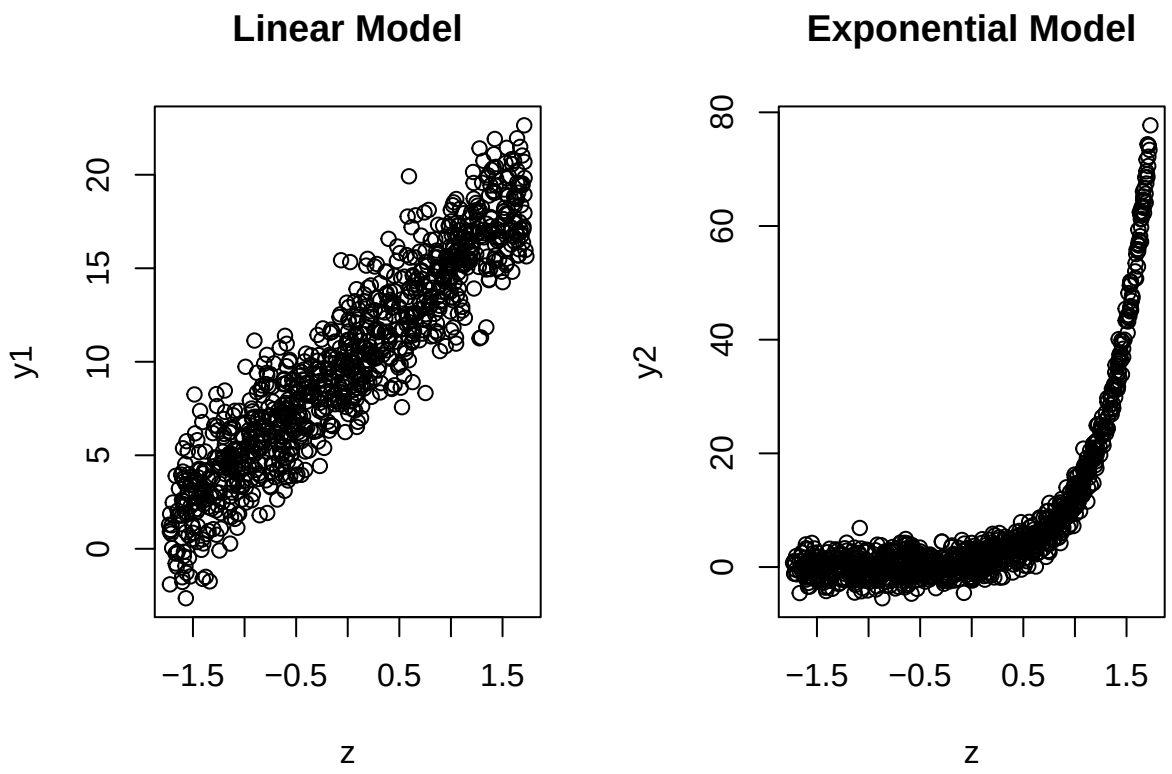
5. Monte Carlo Measures

```
# Matrix that contains the true values ty1 and ty2
Aty1    =  array(data = 0, dim = c(3, 4, K))        # dim = 3 * 4 * K
Aty2    =  array(data = 0, dim = c(3, 4, K))
for (l in 1:3)
  {
  for (c in 1:4)
    {
    Aty1[l,c,]  = ty1
    Aty2[l,c,]  = ty2
    }
}
# Computation of the error for the 6 estimators
error_y1c       = ty1c - Aty1
error_y2c       = ty2c - Aty2
error_y1n       = ty1n - Aty1
error_y2n       = ty2n - Aty2
error_y1conv    = ty1conv - Aty1
error_y2conv    = ty2conv - Aty2
# Relative Error
R_error_y1c     = 100 * (ty1c / Aty1 - 1)
R_error_y2c     = 100 * (ty2c / Aty2 - 1)
R_error_y1conv= 100 * (ty1conv / Aty1 - 1)
R_error_y2conv= 100 * (ty2conv / Aty2 - 1)
R_error_y1n     = 100 * (ty1n / ty1 - 1)
R_error_y2n     = 100 * (ty2n / ty2 - 1)
# Relative Biase
Rbias_y1c   = 100 * apply(error_y1c, c(1, 2), mean) / mean(ty1)
Rbias_y2c   = 100 * apply(error_y2c, c(1, 2), mean) / mean(ty2)
Rbias_y1conv= 100 * apply(error_y1conv, c(1, 2), mean) / mean(ty1)
Rbias_y2conv= 100 * apply(error_y2conv, c(1, 2), mean) / mean(ty2)
Rbias_y1n   = 100 * mean(error_y1n) / mean(ty1)
Rbias_y2n   = 100 * mean(error_y2n) / mean(ty2)
# CV
cv_y1c      = 100 * apply(error_y1c, c(1, 2), sd) / mean(ty1)
cv_y2c      = 100 * apply(error_y2c, c(1, 2), sd) / mean(ty2)
cv_y1conv   = 100 * apply(error_y1conv, c(1, 2), sd) / mean(ty1)
cv_y2conv   = 100 * apply(error_y2conv, c(1, 2), sd) / mean(ty2)
cv_y1n      = 100 * sd(error_y1n) / mean(ty1)
```

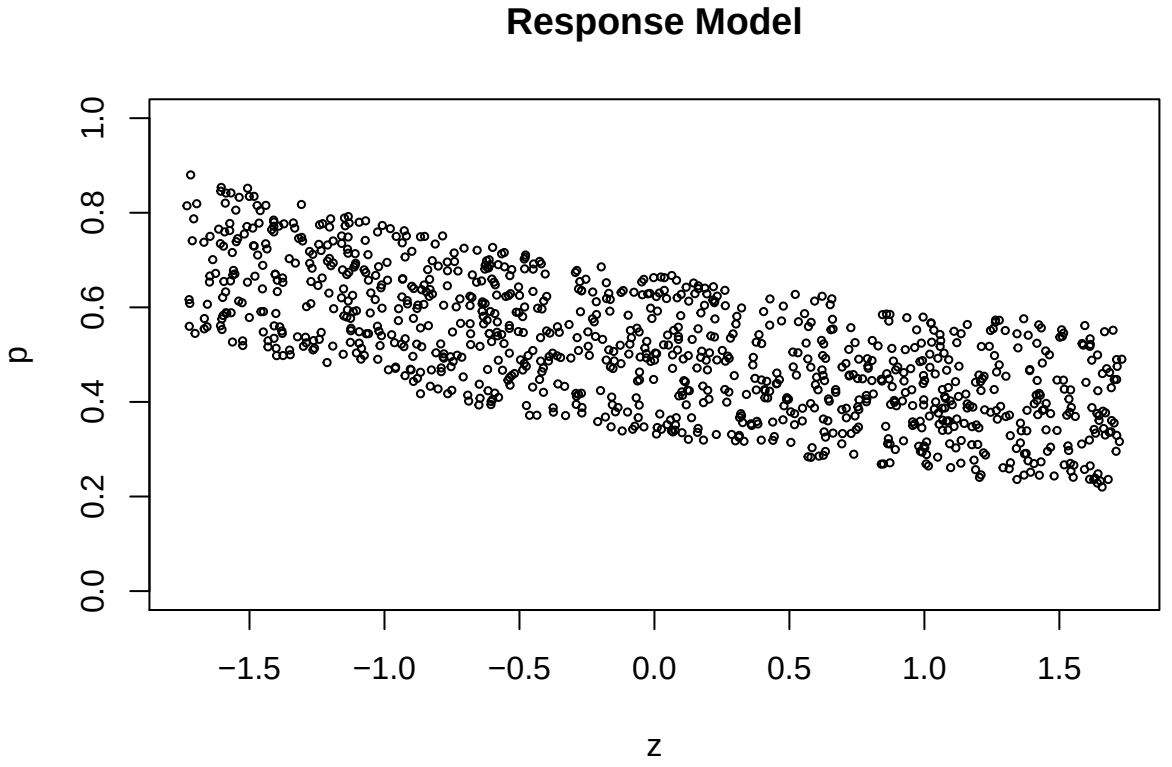
```
cv_y2n      = 100 * sd(error_y2n) / mean(ty2)

# Preparation of the data for the Tables (1)--(4) of the article
tab_y1c     =   rbind(c( 0, 0.1, 0.3, 0.5),
                 Rbias_y1c[1, ], cv_y1c[1,],
                 Rbias_y1c[2, ], cv_y1c[2,],
                 Rbias_y1c[3, ], cv_y1c[3,] )

tab_y2c     =   rbind(c( 0, 0.1, 0.3, 0.5),
                 Rbias_y2c[1, ], cv_y2c[1,],
                 Rbias_y2c[2, ], cv_y2c[2,],
                 Rbias_y2c[3, ], cv_y2c[3,] )

tab_y1conv  =   rbind(c( 0, 0.1, 0.3, 0.5),
                    Rbias_y1conv[1, ], cv_y1conv[1,],
                    Rbias_y1conv[2, ], cv_y1conv[2,],
                    Rbias_y1conv[3, ], cv_y1conv[3,] )

tab_y2conv  =   rbind(c( 0, 0.1, 0.3, 0.5),
                    Rbias_y2conv[1, ], cv_y2conv[1,],
                    Rbias_y2conv[2, ], cv_y2conv[2,],
                    Rbias_y2conv[3, ], cv_y2conv[3,] )
```

6. Figures and Tables of the article

```
# Figure 2: Graphics of y1 and y2 as a function of z
layout(matrix(c(1,2), 1, 2, byrow = TRUE))
plot(z, y1, xlab = "z", ylab = "y1", main ="Linear Model")
plot(z, y2, xlab = "z", ylab = "y2", main ="Exponential Model")
```




```
# Figure 3: Graphic of p as function of z
layout(1)
plot(sqrt(3)*seq(-1,1,length=10), seq(0,1,length=10),  type = "n",
     xlab = "z", ylab = "p", main = "Response Model" )
points(z, p, col=1, cex=0.5)
```



```r
# Tables
format(tab_y1c, nsmall = 1, digits = 2  )          # Table 1
```

```
##      [,1]      [,2]      [,3]      [,4]
## [1,] "  0.000" "  0.100" "  0.300" "  0.500"
## [2,] "  0.036" " -1.650" " -4.932" " -8.071"
## [3,] "  2.210" "  2.200" "  2.115" "  2.075"
## [4,] "  0.138" " -2.393" " -7.326" "-12.009"
## [5,] "  3.714" "  3.662" "  3.561" "  3.436"
## [6,] "  0.477" " -4.660" "-14.540" "-23.587"
## [7,] "  8.617" "  8.061" "  8.004" "  8.188"
```

```r
format(tab_y2c, nsmall = 1, digits = 2  )              # Table 2
```

```
##      [,1]      [,2]      [,3]      [,4]
## [1,] "  0.000" "  0.100" "  0.300" "  0.500"
## [2,] " -0.044" " -4.015" "-11.746" "-19.138"
## [3,] "  6.823" "  6.807" "  6.425" "  6.197"
## [4,] "  0.191" " -5.770" "-17.382" "-28.415"
## [5,] "  9.826" "  9.571" "  9.249" "  8.837"
## [6,] "  0.969" "-11.116" "-34.382" "-55.706"
## [7,] " 20.791" " 19.404" " 19.196" " 19.660"
```

```r
format(tab_y1conv, nsmall = 1, digits = 2   )          # Table 3
```

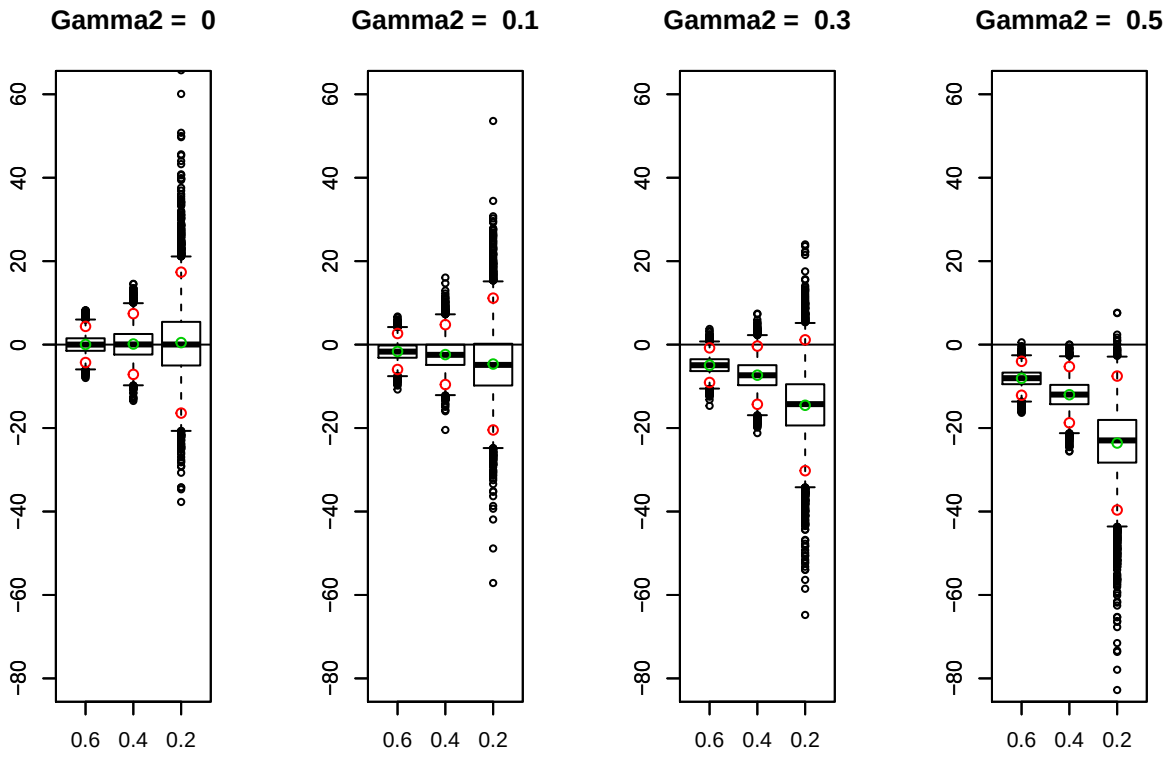
```
##      [,1]   [,2]   [,3]   [,4]
## [1,] " 0.0" " 0.1" " 0.3" " 0.5"
## [2,] "-5.7" "-6.3" "-7.5" "-8.6"
## [3,] " 1.4" " 1.4" " 1.4" " 1.4"
## [4,] "-7.5" "-7.9" "-8.7" "-9.5"
## [5,] " 1.5" " 1.5" " 1.5" " 1.5"
## [6,] "-8.6" "-8.8" "-9.2" "-9.6"
## [7,] " 1.6" " 1.6" " 1.6" " 1.6"
```

```r
format(tab_y2conv, nsmall = 1, digits = 2   )          # Table 4
```

```
##      [,1]    [,2]    [,3]    [,4]
## [1,] "  0.0" "  0.1" "  0.3" "  0.5"
## [2,] "-13.6" "-15.0" "-17.7" "-20.4"
## [3,] "  5.5" "  5.5" "  5.3" "  5.2"
## [4,] "-17.8" "-18.7" "-20.5" "-22.4"
## [5,] "  5.7" "  5.6" "  5.5" "  5.4"
## [6,] "-20.3" "-20.8" "-21.7" "-22.7"
## [7,] "  5.7" "  5.7" "  5.6" "  5.6"
```

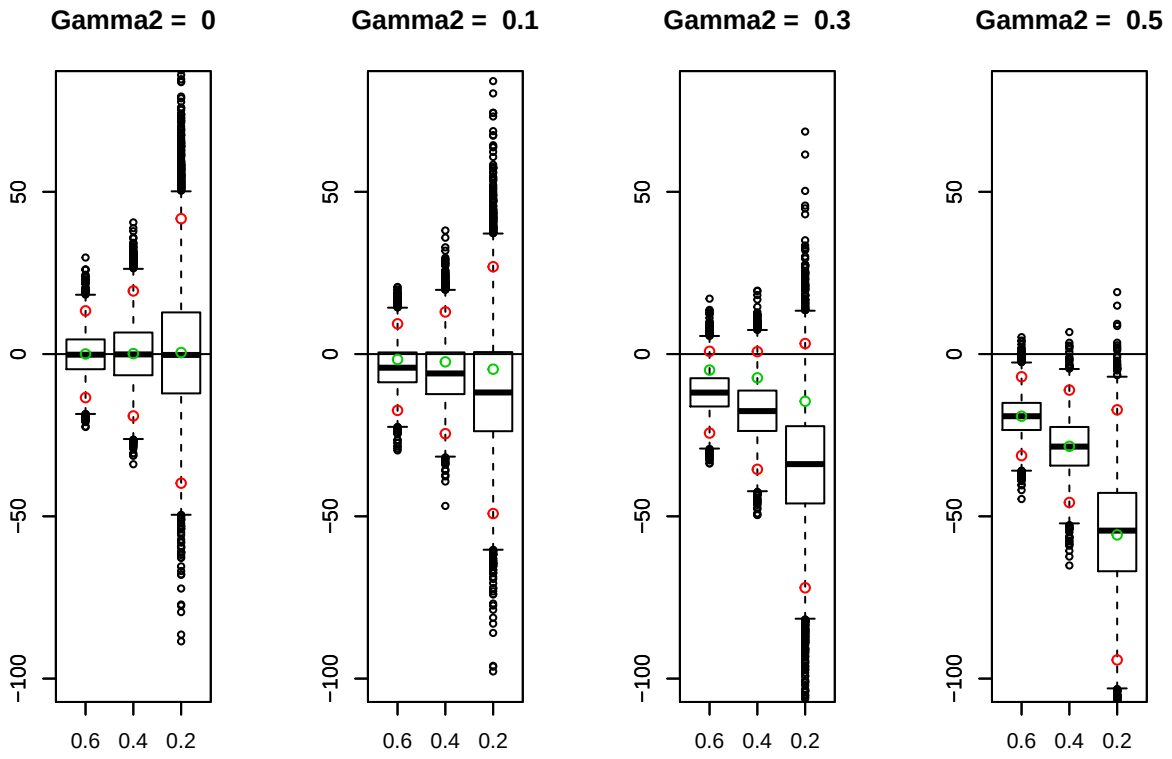
```r
# Figure 4: box-plots of the instrumental calibration estimators ty1c
layout(matrix(1:4, 1, 4,byrow=TRUE))
yech = 60
i = 1
boxplot(R_error_y1c[1,i,], R_error_y1c[2,i,],R_error_y1c[3,i,],
        ylim = c(-yech-20, yech),
        main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y1c[,i]  +1.96* cv_y1c[,i],col=2)
```

```
points(Rbias_y1c[,i]  -1.96* cv_y1c[,i],col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()

i = 2
boxplot(R_error_y1c[1,i,], R_error_y1c[2,i,],R_error_y1c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y1c[,i]  +1.96* cv_y1c[,i],col=2)
points(Rbias_y1c[,i]  -1.96* cv_y1c[,i],col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()

i = 3
boxplot(R_error_y1c[1,i,], R_error_y1c[2,i,],R_error_y1c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y1c[,i]  +1.96* cv_y1c[,i],col=2)
points(Rbias_y1c[,i]  -1.96* cv_y1c[,i],col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()
i = 4
boxplot(R_error_y1c[1,i,], R_error_y1c[2,i,],R_error_y1c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y1c[,i]  +1.96* cv_y1c[,i],col=2)
points(Rbias_y1c[,i]  -1.96* cv_y1c[,i],col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()
```




```
# Figure 5: box-plots of the instrumental estimators ty2c
layout(matrix(1:4, 1, 4, byrow=TRUE))
yech = 80
i = 1
boxplot(R_error_y2c[1,i,], R_error_y2c[2,i,],R_error_y2c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y2c[,i]  +1.96* cv_y2c[,i],col=2)
points(Rbias_y2c[,i]  -1.96* cv_y2c[,i],col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()

i = 2
boxplot(R_error_y2c[1,i,], R_error_y2c[2,i,],R_error_y2c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i],col=3)
points(Rbias_y2c[,i] + 1.96* cv_y2c[,i], col=2)
points(Rbias_y2c[,i] - 1.96* cv_y2c[,i], col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()

i = 3
```

```
boxplot(R_error_y2c[1,i,], R_error_y2c[2,i,],R_error_y2c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y1c[,i], col=3)
points(Rbias_y2c[,i] + 1.96* cv_y2c[,i], col=2)
points(Rbias_y2c[,i] - 1.96* cv_y2c[,i], col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()
i = 4
boxplot(R_error_y2c[1,i,], R_error_y2c[2,i,],R_error_y2c[3,i,],
        ylim = c(-yech-20, yech)
        ,main = paste("Gamma2 = ", Gamma_2[i])
)
abline(h = 0)
points(Rbias_y2c[,i], col=3)
points(Rbias_y2c[,i] + 1.96 * cv_y2c[,i], col=2)
points(Rbias_y2c[,i] - 1.96 * cv_y2c[,i], col=2)
axis(1, at = 1:3, labels = c(0.6,0.4,0.2) )
axis(2)
box()
```



A cautionary tale on instrument vector calibration for the treatment of nonignorable unit nonresponse in surveys

Eric Lesage, David Haziza and Xavier D'Haultfoeuille

July 5, 2017

R code corresponding to the empirical investigation on the variance estimatior presented in Section 5 of the article

Supplementary material

1. Function used to generate the proxy variables x

```
gen_proxy<- function(gamma0, gamma1, gamma2, z, u)
            {
            sigmax=(1 - gamma1^2 - gamma2^2)^(0.5)
            return(gamma0 * rep(1, N) + gamma1 * z + gamma2 * u
            + sigmax * rnorm(n = N, mean=0, sd=1))
            }
```

2. Parameters

```
# 1. Population size and number of iterations
set.seed(2017)        # seed used to find exactly the same results
                      # each time the simulation is performed
N   = 50000             # Population size
n   = 1500             # Sample size
K   = 10000 # Number of Monte Carlo replicates

# 2. Values of the paramaters gamma_1 and gamma_2 for the proxy variables
Gamma_0     = 0
Gamma_1     = c(0.6, 0.4, 0.2)              # dim = 3
Gamma_2     = c(0.0, 0.1, 0.3, 0.5 )        # dim = 4
```

3. Variables format

```
# Variables
X     = array(data = 0, dim = c(3, 4, N))           # dim = 3 * 4 * N
y1  = rep(0, N)           # First variable of interest
y2  = rep(0, N)           # Second variable of interest
one = rep(1, N)             # Vector of 1, dummy variable
```

```
R   = rep(0, N)           # Indicator variable on nonresponse

# True values and estimators
# ty1: true total
ty1     = rep(0, K)
# ty1c: instrument vector calibration estimator
ty1c    = array(data = 0, dim = c(3, 4, K))                    # dim = 3 * 4 * K
# Variance estimator of the instrument vector calibration estimator
v.hat.1c = array(data = rep(0, 3*4*K), dim = c(3, 4, K))       # dim = 3 * 4 * K

# The same for the variable Y2
ty2       = rep(1, K)
ty2c       = array(data = 0, dim = c(3, 4, K))                 # dim = 3 * 4 * K
v.hat.2c  = array(data = rep(0, 3*4*K), dim = c(3, 4, K))      # dim = 3 * 4 * K
```

4. Variance Estimation

4.1 Replicates

```
library(sampling)   # We use the function gencalib() of
                    # the package sampling to get the calibration weights
# Generation of the variables U and Z
z = sqrt(3) * runif(n=N, min = -1 , max = 1)
u = sqrt(3) * runif(n=N, min = -1 , max = 1)
# Generation of the variable Y_1k: Linear model
y1 = 10 + 5 * z + rnorm(n=N, mean=0, sd=2)
# Generation of the variable Y_2k: Exponential model
y2 = exp(2.5*z) + rnorm(n=N, mean=0, sd=2)
# We performe the  K = 10 000 replicates
for (j in 1:K)  {
  ## Sampling: SRS
  I = rep(0, N)
  I[sample(N, n)] = 1
  # Generation of the variables p and R with respect of the Nonresponse model
  p = 1 / (2 + 0.35 * z)  +  0.1 * u
  R = rbinom(n = N, size = 1, prob = p)
  ## Sample of respondents
  R[!I] =  0
  # Finite population parameters: t_y1 and t_y2
  ty1[j]  = sum(y1)
  ty2[j]  = sum(y2)
  ## Estimators
  for (l in 1:3)    {
    for (c in 1:4)  {
      # Generation of the variables 12 X(gamma_1, gamma_2)
      X[l,c,] = gen_proxy(Gamma_0, Gamma_1[l] , Gamma_2[c], z, u)
      ## Computation of the 12 Instrument-vector calibration estimators
      total = c(N, sum(X[l,c,]))
      Xsr   = cbind(one[(R==1)], X[l,c,(R==1)])
      Zsr   = cbind(one[(R==1)], z[(R==1)])
      Fcal  = gencalib( Xsr, Zsr,
```

```
                        d = N / n * one[(R==1)], total, method = "linear")
      ty1c[l,c,j]   = N/n * t(Fcal) %*% y1[(R==1)]
      ty2c[l,c,j]   = N/n * t(Fcal) %*% y2[(R==1)]
      # Regression coefficients associated to the 12 Instrument-vector calibration estimators
      B.hat1c  = solve(t(Zsr) %*% Xsr) %*%  (t(Zsr) %*% y1[(R==1)])
      B.hat2c  = solve(t(Zsr) %*% Xsr) %*%  (t(Zsr) %*% y2[(R==1)])
      # Linearized variables
      eta1c    = ( y1[(R==1)] - Xsr %*% B.hat1c) * Fcal
      eta2c    = ( y2[(R==1)] - Xsr %*% B.hat2c) * Fcal
      eta1c   = c(eta1c, rep(0,n - length(eta1c)))
      eta2c   = c(eta2c, rep(0,n - length(eta2c)))
      # Linearized variances associated to the 12 Instrument-vector calibration estimators
      v.hat.1c[l,c,j] = N^2 * (1-n/N) * (var(eta1c)) /n
      v.hat.2c[l,c,j] = N^2 * (1-n/N) * (var(eta2c)) /n
    }
  }
}
```

4.2 Monte Carlo Measures

```
# Y1: linear model
var_y1c_MC     = apply(ty1c, c(1, 2), var)
var_y1c_hat.MC = apply(v.hat.1c, c(1, 2), mean)
# Monte Carlo Variance - Y1 - Instrument Vector Calibration
format(var_y1c_MC, big.mark = " ", scientific = T)
```

```
##      [,1]           [,2]           [,3]           [,4]
## [1,] "1.591405e+08" "1.578487e+08" "1.509417e+08" "1.466913e+08"
## [2,] "4.661638e+08" "4.370347e+08" "4.285664e+08" "4.041799e+08"
## [3,] "2.242198e+09" "2.060582e+09" "1.925885e+09" "1.977512e+09"
```

```
# Estimated Variance (MC mean) - Y1 - Instrument Vector Calibration
format(var_y1c_hat.MC, big.mark = " ", scientific = T)
```

```
##      [,1]           [,2]           [,3]           [,4]
## [1,] "1.587036e+08" "1.551770e+08" "1.493461e+08" "1.444430e+08"
## [2,] "4.514418e+08" "4.388049e+08" "4.173593e+08" "4.034492e+08"
## [3,] "2.291735e+09" "2.141111e+09" "2.002883e+09" "2.029252e+09"
```

```
# Monte Carlo Relative Bias - Y1 - Instrument Vector Calibration
RB_1_IV = 100* (var_y1c_hat.MC - var_y1c_MC) / var_y1c_MC
format( 100* (var_y1c_hat.MC - var_y1c_MC) / var_y1c_MC , digits = 2,  scientific = F)
```

```
##      [,1]    [,2]    [,3]    [,4]
## [1,] "-0.27" "-1.69" "-1.06" "-1.53"
## [2,] "-3.16" " 0.41" "-2.62" "-0.18"
## [3,] " 2.21" " 3.91" " 4.00" " 2.62"
```

```
# Y2: Exponential model
var_y2c_MC= apply(ty2c, c(1, 2), var)
var_y2c_hat.MC= apply(v.hat.2c, c(1, 2), mean)
# Monte Carlo Variance - Y2 - Instrument Vector Calibration
format(var_y2c_MC, big.mark = " ", scientific = T)
```

```
##      [,1]           [,2]           [,3]           [,4]
```

```
## [1,] "1.120708e+09" "1.075501e+09" "9.907684e+08" "9.166729e+08"
## [2,] "2.422575e+09" "2.251125e+09" "2.124074e+09" "1.968177e+09"
## [3,] "9.989157e+09" "9.199191e+09" "8.471435e+09" "8.539925e+09"
```

```
# Estimated Variance (MC mean) - Y2 - Instrument Vector Calibration
format(var_y2c_hat.MC, big.mark = " ", scientific = T)
```

```
##      [,1]           [,2]           [,3]           [,4]
## [1,] "1.109891e+09" "1.062327e+09" "9.800715e+08" "9.108822e+08"
## [2,] "2.363086e+09" "2.262754e+09" "2.090437e+09" "1.967267e+09"
## [3,] "1.022452e+10" "9.507593e+09" "8.779775e+09" "8.813206e+09"
```

```
# Monte Carlo Relative Bias - Y2 - Instrument Vector Calibration
RB_2_IV = 100* (var_y2c_hat.MC - var_y2c_MC) / var_y2c_MC
format( 100* (var_y2c_hat.MC - var_y2c_MC) / var_y2c_MC , digits = 2,  scientific = F)
```

```
##      [,1]     [,2]     [,3]     [,4]
## [1,] "-0.965" "-1.225" "-1.080" "-0.632"
## [2,] "-2.456" " 0.517" "-1.584" "-0.046"
## [3,] " 2.356" " 3.352" " 3.640" " 3.200"
```

5. Latex Tables

```
library(xtable)
# Table 1 - Relative bias of the variance estimators of the
# Variable Y1 generated according to the linear model
tabin =  RB_1_IV
colnames(tabin) = c(0.0, 0.1, 0.3, 0.5 )
rownames(tabin) = c( 0.6, 0.4,0.2)
tableau = xtable(tabin,
                 caption = "Monte Carlo percent relative bias of the variance estimators,
                 $\\hat{V}_1$, of the instrumental calibration estimator for different pairs
                 $(\\Gamma_1, \\Gamma_2)$ corresponding to population generated according
                 to (\\ref{lin_mod})",
                 align   = "|r|r|r|r|r|",
                 digits  = 1,
                 label   = "tab.RB.IVC.lin")
print(tableau,
      file="tables_2.tex",
      append = FALSE,
      hline.after=c(-1, 0, 1, 2, 3), sanitize.text.function = function(x){x})

# Table 2 - Relative bias of the variance estimators of the
# Variable Y2 generated according to the exponential model
tabin =  RB_2_IV
colnames(tabin) = c(0.0, 0.1, 0.3, 0.5 )
rownames(tabin) = c( 0.6, 0.4,0.2)
tableau = xtable(tabin,
                 caption = "Monte Carlo percent relative bias of the variance estimators,
                 $\\hat{V}_1$, of the instrumental calibration estimator for different pairs
                 $(\\Gamma_1, \\Gamma_2)$ corresponding to population generated according
                 to (\\ref{exp_mod})",
                 align   = "|r|r|r|r|r|",
```

```
                digits  = 1,
                label   = "tab.RB.IVC.exp")
print(tableau,
      file="tables_2.tex",
      append = TRUE,
      hline.after=c(-1, 0, 1, 2, 3), sanitize.text.function = function(x){x})
```