Results from Y as Normal and (X,Z,W) as a truncated mixture

Contents

Definition of Parameters

The causal effect is $\Delta = E(Y(1) - Y(0))$. We define a logistic regression model for the propensity score $p = P(T = 1)$, where T is the indicator of treatment assigned, $f(T \mid X, Z, \psi) \sim Bernoulli(expit(\gamma_0 + \gamma_X X + \gamma_z Z))$.

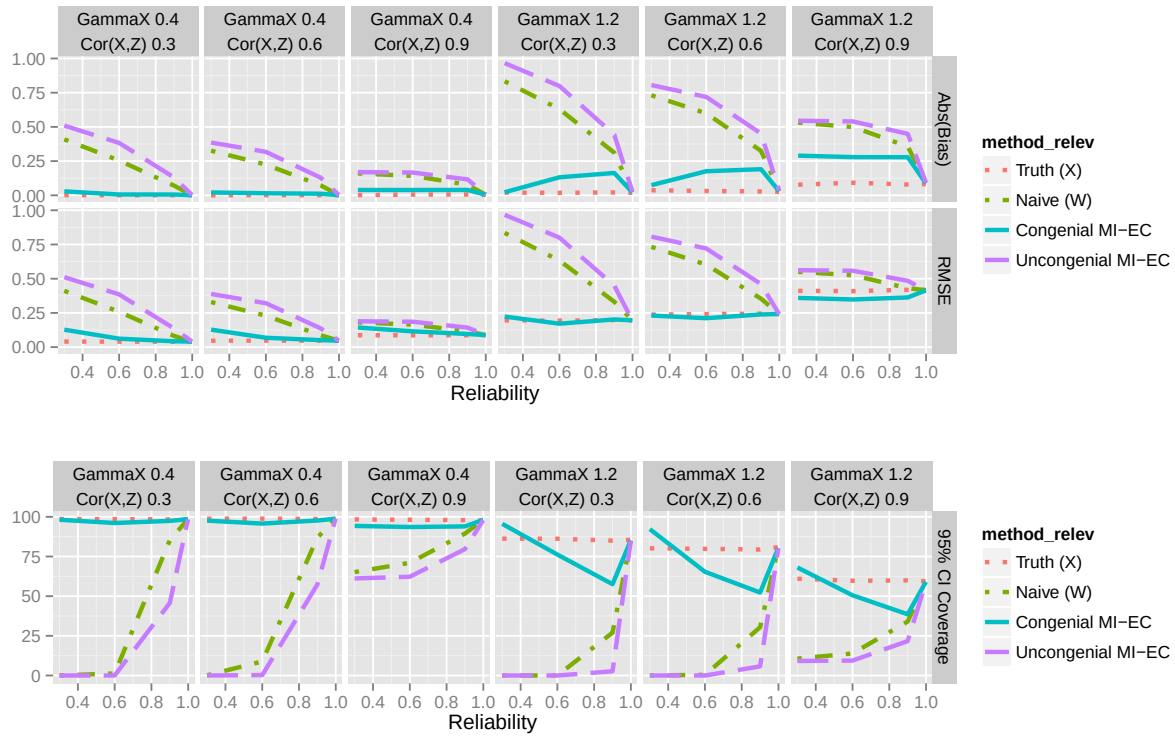
We generate X, Z as follows. We construct X with $f(X_1) \sim F(200, 100), \quad f(X_2) \sim \chi^2(1), \quad f(X_3) \sim N(0, 1)$, and define $X = (X_1 + -2X_2 + \frac{1}{2}X_3)$ if $X > -4$, else $X = 4$. We then generate Z to have the same distribution and $cor(X, Z) = \rho$.

We use the measurement error model as $f(W \mid T, X, Z) \sim N(\beta_0 + \beta_1 X, \sigma^2)$, and a distribution of the potential outcomes as $f(Y(T) \mid T, X, Z) \sim N(\Delta T + \delta_X X + \delta_Z Z, \tau^2)$.

The original values used in their simulation are:

Distribution	Parameter	Value	Distribution	Parameter	Value
Simulation	N_{sim}	5000	W	β_0	0
	n_{calib}	500		β_1	1
	n_{main}	2500		small σ^2	0.111
T	γ_0	0		moderate σ^2	0.667
	γ_Z	0.4		large σ^2	2.333
	small γ_X	0.4	Y	Δ	2
	large γ_X	1.2		δ_X	0.5
(X, Z)	low ρ	0.3		δ_Z	0.1
	medium ρ	0.6		τ^2	1
	high ρ	0.9			

Figures



Code for simulation

```
#################################
##  Simulation of propensity score with
##   covariates measured with error

##  This code was based on 'Y_normal_XWZ_mixture.R', which
##  was purled from Y_normal_XWZ_mixture.Rmd,
##  and modified to be run on Enigma:
##   - Adds intro, and command to import the iteration from
##     the terminal 'i'
##   - Changes location of sourced file
##   - Changes number of simulations to 50
##   - Changes seed to XXX+i
##   - Eliminates the part that creates the table
##   - Changes the name of the saved results Rdata file

##  This simulation includes the following changes:
##   - Added reliabilities of 0.3, 0.6
##   - Uses m=12, n=3  (as all the previous simulations)

##  It also does:
##   - Runs reliabilities 0.9 and 0.999
##   - Modified the 'sampling' function to also run an 'extra small' variance (with reliability = 0.99)
##   - Eliminates CC, RP. Only calculates naive.
```

```
##   - Uses 4 types of MIEC
##   - Sets reliabilities instead of variances
##   - Uses both a correct model (uses X and Z in the
##      estimation model) as well as IPTW


##  We use this code to evaluate how sensitive MIEc is to the distributional assumptions. To do that
##   - We assume that Y has a normal distribution and
##   - We assume that X comes from a truncated mixted distribution
##   - We add the library "ecodist" (this allows the creation of variables with certain correlation)

##  We also changed the parameter Delta, the treatment effect, to 2. This reduces the unbalances in Y (w
##  the values for Y1 were 1, in the bernoulli case)

##  We update the definition of reliability




temp <- commandArgs(TRUE)
i    <-as.numeric(temp[1])

## ----load_libraries, message=FALSE, warning=FALSE------------------------
library(mvtnorm)
library(mitools)
library(MCMCpack)
library(xtable)
library(survey)
library(ecodist)
#library(Gmisc, verbose=FALSE) #verbose option does nothing when using knitr
source("MI-EC algorithm.r")

## ----def_params----------------------------------------------------------
# Sample size and number of simulations
n_main  <- 2500
n_calib <- 500
N_sim   <- 50

# Distribution of T | X,Z
gamma_0  <- 0 #This will determine the proportion receiving treatment
gamma_Z  <- 0.4
gamma_Xs <- 0.4
gamma_Xl <- 1.2

# Distribution of (X,Z)
rho_l    <- 0.3
rho_m    <- 0.6
rho_h    <- 0.9


# Distribution of W | Y,X,Z
beta_0   <- 0
beta_1   <- 1
```

```r
#The variance of X is obtain by doing
# X =  rf(10000000,200,100)-2*rchisq(10000000,1)+0.5*rnorm(10000000,0,1)
# X[X< -4]=4
# var(X)

var_x     = 3.39
rels      <- c(0.3,0.6,0.9,0.999)
sigmas    <- var_x*((1/rels) - beta_1^(2))
sigma2_xs <- sigmas[4]
sigma2_s  <- sigmas[3]
sigma2_m  <- sigmas[2]
sigma2_l  <- sigmas[1]

# Distribution of Y(T) | T, X, Z
Delta    <- 2
delta_X  <- 0.5
delta_Z  <- 0.1
tau2     <- 1


## ----expitlogit-----------------------------------------------------------
expit <- function(x) exp(x)/(1+exp(x))
logit <- function(p) log(p/(1-p))


## ----sampling_fun---------------------------------------------------------
sampling <- function(cor_level, X_effect, m_error){

X =  rf(n_main+n_calib,200,100)-2*rchisq(n_main+n_calib,1)+0.5*rnorm(n_main+n_calib,0,1)
X[X< -4]=4


if(cor_level == "low"){
  X_Zl = corgen(n_main+n_calib,X, rho_l, epsilon=0)
}else if(cor_level == "med") {
  X_Zl = corgen(n_main+n_calib,X, rho_m, epsilon=0)
}else if(cor_level == "high") {
  X_Zl = corgen(n_main+n_calib,X, rho_h, epsilon=0)
}else {stop("Correlation level must be 'high', 'med', or 'low'")}

X_Z = matrix(NA, ncol = 2, nrow=n_main+n_calib)
X_Z[,1] = X_Zl[[1]]
X_Z[,2] = X_Zl[[2]]

colnames(X_Z) <- c("X", "Z")



    # Sample Y(0), Y(1) \mid X,Z

    Y0 <- rnorm(n_main+n_calib, mean=(delta_X*X_Z[,"X"] + delta_Z*X_Z[,"Z"]),
                sqrt(tau2))
    Y1 <- rnorm(n_main+n_calib, mean=(Delta + delta_X*X_Z[,"X"] + delta_Z*X_Z[,"Z"]), sqrt(tau2))
```

```
    #Y1 <- Y0 + Delta     # rank preserving


    # Sample from the distribution of $T \mid X,Z,\psi$

    if(X_effect == "small"){
       logit_T <- gamma_0 + gamma_Xs*X_Z[,"X"] + gamma_Z*X_Z[,"Z"]
     } else if(X_effect == "large") {
       logit_T <- gamma_0 + gamma_Xl*X_Z[,"X"] + gamma_Z*X_Z[,"Z"]
     } else {stop("Effect of X must be 'small' or 'large'")}

    T <- rbinom(n_main+n_calib, 1, p= expit(logit_T))


    # Creating Y_obs given Y(0), Y(1), T

    Y_obs <- Y1*T + Y0*(1-T)


    #Sample from the distribution of $W \mid T,X,Z$

    if(m_error == "small"){
        W <- rnorm(n_main+n_calib, mean= (beta_0+beta_1*X_Z[,"X"]), sd=sqrt(sigma2_s))
     } else if(m_error == "moderate") {
        W <- rnorm(n_main+n_calib, mean= (beta_0+beta_1*X_Z[,"X"]), sd=sqrt(sigma2_m))
     } else if(m_error == "large") {
        W <- rnorm(n_main+n_calib, mean= (beta_0+beta_1*X_Z[,"X"]), sd=sqrt(sigma2_l))
     } else if(m_error == "extra small") {
        W <- rnorm(n_main+n_calib, mean= (beta_0+beta_1*X_Z[,"X"]), sd=sqrt(sigma2_xs))
     } else {stop("Measurement error must be 'extra small', 'small', 'moderate', or 'large'")}


    #Setting datasets for calibration and for main inference

    i.calib <- 1:n_calib
    i.main  <- (n_calib+1):(n_calib+n_main)

    data.main  <- data.frame(W = W[i.main], T = T[i.main], Z=X_Z[i.main,"Z"],
                             Y_obs = Y_obs[i.main])
    data.cause <- data.frame(p= expit(logit_T[i.main]), X= X_Z[i.main,"X"], Y0 = Y0[i.main], Y1 = Y1[i.m
    data.calib <- data.frame(X= X_Z[i.calib,"X"], W = W[i.calib])

    return(list(main=data.main, calib=data.calib, cause=data.cause))
    }


## ----Correction_functions--------------------------------------------------
true_regression <- function(data_main, Xtrue){
  data_main$Xtrue <- Xtrue

  model <- glm(T~Xtrue+Z, data=data_main, family=binomial)
  p_hat <- predict(model,type="response")
```

```
  data_main$wt <- ifelse(data_main$T==1, 1/p_hat, 1/(1-p_hat))

  design.ate   <- svydesign(ids=~1, weights=~wt, data=data_main)
  survey.model <- svyglm(Y_obs~T, design=design.ate)
  coef         <- summary(survey.model)$coeff["T",c("Estimate", "Std. Error")]

  CI_low <- coef["Estimate"] + qnorm(0.025)*coef["Std. Error"]
  CI_upp <- coef["Estimate"] + qnorm(0.975)*coef["Std. Error"]

  coef <- cbind(coef["Estimate"],coef["Std. Error"],CI_low, CI_upp)
  colnames(coef) <- c("results", "se", "(lower", "upper)")

  return(coef)
}


naive_regression <- function(data_main){
  model <- glm(T ~ W + Z, data=data_main, family=binomial)
  p_hat <- predict(model,type="response")

  data_main$wt <- ifelse(data_main$T==1, 1/p_hat, 1/(1-p_hat))

  design.ate   <- svydesign(ids=~1, weights=~wt, data=data_main)
  survey.model <- svyglm(Y_obs~T, design=design.ate)
  coef         <- summary(survey.model)$coeff["T",c("Estimate", "Std. Error")]

  CI_low <- coef["Estimate"] + qnorm(0.025)*coef["Std. Error"]
  CI_upp <- coef["Estimate"] + qnorm(0.975)*coef["Std. Error"]

  coef <- cbind(coef["Estimate"],coef["Std. Error"],CI_low, CI_upp)
  colnames(coef) <- c("results", "se", "(lower", "upper)")

  return(coef)
}


## ----MIEC_funs-----------------------------------------------------------
multiple_imputation_EC   <- function(data_main, data_calib, option="Ycov"){
  #Option: select 'Ycov', 'noT', 'noY', 'noT'
  # Ycov (Y covariate) uses $T$ as outcome, $Z$ and $Y_{obs}$ as helpful covariates
  # noY (no Y) uses $T$ as outcome, $Z$ as helpful covariate
  # noT (no T) uses $Y_{obs}$ as outcome, $Z$ as helpful covariate
  # noTY (no T, nor Y) uses no outcome, $Z$ as helpful covariate

  # Other parameters for MIEC/ two-stage imputation procedure
  m <- 12 #Number of draws from parameter distribution
  n <- 3  #Number of samples (and imputations) for each m

  # Generating Multiple Imputations:

  if(option=="Yout"){ # Y as a helpful covariate
    q <- 2  #Dimension of T. T = (T,Y_obs)
    r <- 1  #Dimension of Z. Z = (Z)
```

```
  MIEC_data <- MIEC(data_main[,c("W","T","Y_obs","Z")],data_calib,n_calib,n_main,M=m,N=n,K=q,S=r)

}else if(option=="noY"){
  q <- 1  #Dimension of T. T = T
  r <- 1  #Dimension of Z. Z = Z
  MIEC_data <- MIEC(data_main[,c("W","T","Z")],data_calib,n_calib,n_main,M=m,N=n,K=q,S=r)

}else if(option=="noT"){
  q <- 1  #Dimension of T. T = Y
  r <- 1  #Dimension of Z. Z = Z
  MIEC_data <- MIEC(data_main[,c("W","Y_obs","Z")],data_calib,n_calib,n_main,M=m,N=n,K=q,S=r)

}else if(option=="noTY"){
  q <- 0  #Dimension of T. T = NULL
  r <- 1  #Dimension of Z. Z = Z
  MIEC_data <- MIEC(data_main[,c("W","Z")],data_calib,n_calib,n_main,M=m,N=n,K=q,S=r)

}else {stop("Only options 'Ycov', 'noT', 'noY', 'noT' are accepted")}


imputed_cols <- (q+r+1):ncol(MIEC_data)

delta_MI <- matrix(NA,ncol=length(imputed_cols),nrow=2)

MI_data  <- data.frame(
  T = data_main[,"T"],
  Z = data_main[,"Z"],
  Y_obs = data_main[,"Y_obs"])

for(k in imputed_cols){
  MI_data$X <- MIEC_data[,k]
  model <- glm(T ~ X + Z, data=MI_data, family=binomial)
  p_hat <- predict(model,type="response")

  MI_data$wt <- ifelse(data_main$T==1, 1/p_hat, 1/(1-p_hat))

  design.ate   <- svydesign(ids=~1, weights=~wt, data=MI_data)
  survey.model <- svyglm(Y_obs~T, design=design.ate)
  delta_MI[,k-(q+r)] <- summary(survey.model)$coeff["T",c("Estimate", "Std. Error")]
}


estimate.mitools <- summary(MIcombine(results = as.list(delta_MI[1,]),
                                      variances = as.list(delta_MI[2,]^2)))[,-5]


combine_foo <- function(coef, vars){

  # coefficient of interest (gamma_x_hat)
  gamma_hat_MI <- mean(coef)

  # The following code gives non-sensical results
  #Calculating W,B,U
```

```
    gamma_matrix <- matrix(coef, ncol=n, byrow=TRUE)
    mean_n_gamma_hat <- apply(gamma_matrix, 1, mean)

    W <- sum(sapply(1:m, function(x){
      sum((gamma_matrix[x,] - mean_n_gamma_hat[x])^2)
    }))/(m*(n-1))
    B <- sum((mean_n_gamma_hat - gamma_hat_MI)^2)/(m-1)

    U <- mean(vars)

    # Calculating variance of our coefficient of interest
    T_MI <- U - W + (1+1/m)*B - W/n
    if(T_MI < 0) T_MI <- (1+1/m)*B

    # Confidence intervals are done with t-distribution with these
    # degrees of freedom
    df <- 1/((((1+1/m)*B)^2)/((m-1)*T_MI^2)) +
               ((((1+1/n)*W)^2)/(m*(n-1)*T_MI^2))) #Note that here was the mistake
    if(T_MI < 0) df <- m-1

    CI_low <- gamma_hat_MI + qt(0.025, df=df)*sqrt(T_MI)
    CI_upp <- gamma_hat_MI + qt(0.975, df=df)*sqrt(T_MI)

    return(c(coef=gamma_hat_MI,se=sqrt(T_MI), CI_low=CI_low, CI_upp=CI_upp))
  }


  estimate.reiter  <- combine_foo(delta_MI[1,], delta_MI[2,]^2)

  return(rbind(estimate.mitools,estimate.reiter))
}



## ----simulation_fun-------------------------------------------------------
full_simulation <- function(cor_level = "low", X_effect = "large", m_error = "large"){
  data <- sampling(cor_level = cor_level, X_effect = X_effect, m_error = m_error)

  Xtrue     <- true_regression(data$main, data$cause$X)
  naive     <- naive_regression(data$main)
  MIEC_Yout <- multiple_imputation_EC(data$main, data$calib, "Yout")
  MIEC_noY  <- multiple_imputation_EC(data$main, data$calib, "noY")
  MIEC_noT  <- multiple_imputation_EC(data$main, data$calib, "noT")
  MIEC_noTY <- multiple_imputation_EC(data$main, data$calib, "noTY")

  result_table      <- rbind(Xtrue, naive,MIEC_Yout, MIEC_noY, MIEC_noT, MIEC_noTY)
  mean_insample     <- mean(data$cause$Y1 - data$cause$Y0)
  result_string     <- c(mean_insample,t(result_table))

  names(result_string) <- c("insample mean",
                            paste0("Xtrue.", colnames(Xtrue)),
                            paste0("naive.", colnames(Xtrue)),
                            paste0("MIEC_Yout_mi.", colnames(Xtrue)),
```

```
                            paste0("MIEC_Yout_re.", colnames(Xtrue)),
                            paste0("MIEC_noY_mi.", colnames(Xtrue)),
                            paste0("MIEC_noY_re.", colnames(Xtrue)),
                            paste0("MIEC_noT_mi.", colnames(Xtrue)),
                            paste0("MIEC_noT_re.", colnames(Xtrue)),
                            paste0("MIEC_noTY_mi.", colnames(Xtrue)),
                            paste0("MIEC_noTY_re.", colnames(Xtrue))
  )

  return(result_string)
}


## ----simulations,cache=TRUE, eval=TRUE-----------------------------------
time1 <- Sys.time()
set.seed(62382+i)

results <- list()

results[["gamma_Xs"]][["sigma2_xs"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "small", m_error = "extra small"))

results[["gamma_Xs"]][["sigma2_xs"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "small", m_error = "extra small"))

results[["gamma_Xs"]][["sigma2_xs"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "small", m_error = "extra small"))


results[["gamma_Xl"]][["sigma2_xs"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "large", m_error = "extra small"))

results[["gamma_Xl"]][["sigma2_xs"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "large", m_error = "extra small"))

results[["gamma_Xl"]][["sigma2_xs"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "large", m_error = "extra small"))




results[["gamma_Xs"]][["sigma2_s"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "small", m_error = "small"))

results[["gamma_Xs"]][["sigma2_s"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "small", m_error = "small"))

results[["gamma_Xs"]][["sigma2_s"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "small", m_error = "small"))


results[["gamma_Xl"]][["sigma2_s"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "large", m_error = "small"))
```

```
results[["gamma_Xl"]][["sigma2_s"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "large", m_error = "small"))

results[["gamma_Xl"]][["sigma2_s"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "large", m_error = "small"))




results[["gamma_Xs"]][["sigma2_m"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "small", m_error = "moderate"))

results[["gamma_Xs"]][["sigma2_m"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "small", m_error = "moderate"))

results[["gamma_Xs"]][["sigma2_m"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "small", m_error = "moderate"))


results[["gamma_Xl"]][["sigma2_m"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "large", m_error = "moderate"))

results[["gamma_Xl"]][["sigma2_m"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "large", m_error = "moderate"))

results[["gamma_Xl"]][["sigma2_m"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "large", m_error = "moderate"))




results[["gamma_Xs"]][["sigma2_l"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "small", m_error = "large"))

results[["gamma_Xs"]][["sigma2_l"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "small", m_error = "large"))

results[["gamma_Xs"]][["sigma2_l"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "small", m_error = "large"))


results[["gamma_Xl"]][["sigma2_l"]][["rho_l"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "low", X_effect = "large", m_error = "large"))

results[["gamma_Xl"]][["sigma2_l"]][["rho_m"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "med", X_effect = "large", m_error = "large"))

results[["gamma_Xl"]][["sigma2_l"]][["rho_h"]] <- sapply(1:N_sim, function(x)
  full_simulation(cor_level = "high", X_effect = "large", m_error = "large"))
time2 <- Sys.time()
```

```
## ----save_results, eval=TRUE-----------------------------------------------
time2-time1

save(results,file=paste("Simulation_Y_normal_XZW_mixture_", i, "-", format(Sys.time(), "%Y%m%d-%H%M"),"
```