R codes for the paper: Chen, Q., Paik, M. C., Kim, M., and Wang, C. (2016). Using link-preserving imputation for logistic partially linear models with missing covariates, *Computational Statistics and Data Analysis*, **101**: 174-185.

In this supplement, we provide a step-by-step illustration on the application of the proposed methods to the NHANES health insurance coverage data. The dataset "data" contains seven variables: y (1: with insurance; 0: without insurance), x_1 (1: healthy; 0: less healthy), x_2 (1: hispanic; 0: non-hispanic), x_3 (1: female; 0: male), x_4 (1: born in US; 0: otherwise), z (age), and r (1: X_1 is observed, 0: X_1 is missing).

Step 1: Obtain $\boldsymbol{\gamma}$ and $\hat{X}^*$ using the link-preserving imputation

```
data$z = (data$z-min(data$z))/(max(data$z)-min(data$z))
comp.data = data[data$r==1,]
incomp.data = data[data$r==0,]
data = rbind(comp.data,incomp.data)
fit1 = glm(x1 ~ x2 + x3 + x4 + y + z, data = comp.data,
        family = binomial(link = "logit"))
gamma = fit1$coeffi
A = gamma[1] + gamma[2]*data$x2 + gamma[3]*data$x3 + gamma[4]*data$x4
        + gamma[6]*data$z
log1 = log(1 + exp(A + gamma[5]))
log0 = log(1 + exp(A))
x1_star = (log1 - log0) / gamma[5]
```

Step 2: Fill in $\hat{X}^*$ and expand the dataset with duplicated records

```
n.obs = nrow(comp.data)
n.miss = nrow(incomp.data)
n = n.obs + n.miss
dup.data = rbind(comp.data,data)
dup.data$x1[(n.obs+1):(n.obs+n)] = x1_star
```

Step 3: Estimate α in the response model

```
fit2 = glm(r ~ x2 + x3 + x4 + y + z, data = data, family = binomial)
pi.all = predict(fit2, type = "response")
pi = pi.all[1:n.obs]
dup.data$weight = c(1/pi, (pi-1)/pi, rep(1,n.miss))
```

Step 4: Estimate μ_X using the complete cases

```
opt_h = n^{-1/5}
X = cbind(dup.data$x1, dup.data$x2, dup.data$x3, dup.data$x4)
kgplm.rs = kgplm(X[1:n.obs,], dup.data$z[1:n.obs], dup.data$y[1:n.obs], h=opt_h,
        grid=data$z, family = "bernoulli", link = "logit", method = c("backfit"),
        weights=dup.data$weight[1:n.obs], kernel= "epanechnikov")
old_beta = kgplm.rs$b
hatvz = kgplm.rs$m.grid
dup.hatvz = hatvz[match(dup.data$z, data$z)]
eta_hat = X[1:n.obs,] %*% old_beta + hatvz[1:n.obs]
w_hat = exp(eta_hat)/(1+exp(eta_hat))^2
mu_x.nume = t(X[1:n.obs,]) %*% diag(c(dup.data$weight[1:n.obs])) %*% w_hat
mu_x.deno = sum(w_hat*dup.data$weight[1:n.obs])
mu_x = mu_x.nume / mu_x.deno
```

Step 5: Estimate β and $\nu(Z)$ using Nadaraya-Watson kernel estimator

```
## the easy-to-implement method ##
cX = X - matrix(rep(c(mu_x),nrow(X)),nc=ncol(X),byrow=T)
kgplm.eti = kgplm(cX, dup.data$z, dup.data$y, h = opt_h, grid = data$z, family =
        "bernoulli", link = "logit", b.start = old_beta, method = c("backfit"),
        weights = dup.data$weight, kernel = "epanechnikov")
beta.eti = kgplm.eti$b
nu.eti = kgplm.eti$m
## the improved method ##
wscore = dup.data$weight*(dup.data$y - inv.logit(X %*% old_beta + dup.hatvz))
Un = diag(as.numeric(wscore))%*%cX
ai = rbind(Un[1:n.obs,], matrix(0, nr=n.miss, nc=ncol(cX)))
bi = -Un[(n.obs+1):nrow(Un),]
kappahat = ginv(t(bi)%*%(bi))%*%(t(bi)%*%ai)
cX.ipv = cX
cX.ipv[(1+n.obs):nrow(cX),] = cX[(1+n.obs):nrow(cX),]%*%kappahat
kgplm.ipv = kgplm(cX.ipv, dup.data$z, dup.data$y, h = opt_h, grid = data$z, family
        = "bernoulli", link = "logit", b.start = old_beta, method = c("backfit"),
        weights = dup.data$weight, kernel = "epanechnikov")
beta.ipv = kgplm.ipv$b
nu.ipv = kgplm.ipv$m
```

Step 6: Estimate the asymptotic variance of $\hat{\beta}$

```
asyvar = function(beta, nu, cX) {
  eta = X %*% beta + nu
```

```
  Gamma = t(cX[1:n.obs,]) %*% diag(dup.data$weight[1:n.obs]
        * exp(eta[1:n.obs])/(1+exp(eta[1:n.obs]))^2) %*% X[1:n.obs,]/n
  e = dup.data$y - inv.logit(eta)
  TT = cbind(rep(1,n), data$x2, data$x3, data$x4, data$y, data$z)
  w.vec = diag(as.numeric(e))%*%cX
  score = dup.data$weight * w.vec
  aibi = rbind(score[1:n.obs,], matrix(0,nr=n.miss,nc=4))
        + score[(n.obs+1):(n.obs+n),]
  aibi_bar =  rbind(w.vec[1:n.obs,]-w.vec[(n.obs+1):(2*n.obs),],
        -w.vec[(2*n.obs+1):(n.obs+n),])
  Lamhat = t(aibi_bar) %*% diag(-(1-pi.all)/pi.all*data$r) %*% TT
  Phihat = t(TT) %*% diag( c(pi.all*(1-pi.all)) ) %*% TT
  Psihat = diag(c(data$r - pi.all)) %*% TT
  Xi = t(aibi + Psihat %*% solve(Phihat) %*% t(Lamhat)) %*%
        (aibi + Psihat %*% solve(Phihat) %*% t(Lamhat))/n
  asy.var = diag(solve(Gamma) %*% Xi %*% solve(Gamma)/n)
  return(asy.var)
}
## the easy-to-implement method ##
var.eti = asyvar(beta.eti, nu.eti, cX)
## the improved method ##
var.ipv = asyvar(beta.ipv, nu.ipv, cX.ipv)
```