Supplementary Material:
A Review on Optimal Subsampling Methods for Massive Datasets

YAQIONG YAO[1] AND HAIYING WANG*[1]

[1]*Department of Statistics, University of Connecticut, Storrs, Connecticut, USA*

Code

```
installed = rownames(installed.packages())
if (!("survey" %in% installed)) install.packages(c("survey"))
if (!("quantreg" %in% installed)) install.packages(c("quantreg"))

require("survey")
require("quantreg")
AdpOptSubLog <- function(X, y, r0, r, optmethod, data, covariate){
  N <- length(y)
  n1 <- sum(y)
  n0 <- N - sum(y)
  d <- dim(X)[2]

  PI.prop <- rep(1/(2*n1), N); PI.prop[y==0] <- 1/(2*n0)
  idx.prop <- sample(1:N, r0, T, PI.prop)
  x.prop <- X[idx.prop,]; y.prop <- y[idx.prop]
  pinv.prop <- 1/PI.prop[idx.prop]

  data.s1 <- data[idx.prop,]
  design <- svydesign(ids=~1,
                      weights=~pinv.prop,
                      data=as.data.frame(data.s1))
  beta.prop <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                        covariate)),
                      design=design,
                      family=quasibinomial)$coefficients

  P.prop  <- 1 - 1 / (1 + exp(c(X %*% beta.prop)))
  p.prop <- P.prop[idx.prop]
  phi.prop <- p.prop * (1 - p.prop)

  if(optmethod == "A"){
```

*Corresponding author. Email: haiying.wang@uconn.edu

```
    W.prop <- solve(t(x.prop) %*% (x.prop * phi.prop * pinv.prop))
    PI.opt <- sqrt((y - P.prop)^2 * rowSums((X %*% W.prop)^2))
    PI.opt <- c(PI.opt / sum(PI.opt))
  } else if(optmethod == "L") {
    PI.opt <- sqrt((y - P.prop)^2 * rowSums(X^2))
    PI.opt <- c(PI.opt / sum(PI.opt))
  }
  idx.opt <- sample(1:N, r, T, PI.opt)
  x.opt <- X[c(idx.prop, idx.opt),]; y.opt <- y[c(idx.prop, idx.opt)]
  pinv.opt <- c(pinv.prop, 1/PI.opt[idx.opt])

  data.s1 <- data[c(idx.prop, idx.opt),]
  design <- svydesign(ids=~1, weights=~pinv.opt, data=as.data.frame(data.s1))
  beta.opt <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                       covariate)),
                     design=design,
                     family=quasibinomial)$coefficients

  p.opt  <- 1 - 1 / (1 + exp(c(x.opt %*% beta.opt)))
  phi.opt <- p.opt * (1 - p.opt)
  W.opt <- solve(t(x.opt) %*% (x.opt * phi.opt * pinv.opt))
  V.opt <- t(x.opt) %*% (x.opt * ((y.opt - p.opt) * pinv.opt)^2)
  se.opt <- sqrt(diag(W.opt %*% V.opt %*% W.opt))

  zvalue <- beta.opt/se.opt
  pvalue <- 2*pnorm(-abs(zvalue))
  summary <- data.frame(coefficients = beta.opt, stdErr = se.opt,
                        Zvalue = zvalue, Pvalue = pvalue)
  row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
  return(summary)
}

AdpUnif <- function(X, y, r0, r){
  N <- length(y)
  d <- dim(X)[2]
  unif.idx <- sample(1:N, r0 + r, T)
  x.unif <- X[unif.idx,]
  y.unif <- y[unif.idx]
  beta.unif <- glm(y.unif~x.unif[,-1],
                   family = "binomial")$coefficients

  p.unif  <- 1 - 1 / (1 + exp(c(x.unif %*% beta.unif)))
  phi.unif <- p.unif * (1 - p.unif)
```

```
  W.unif <- solve(t(x.unif) %*% (x.unif * phi.unif))
  V.unif <- t(x.unif) %*% (x.unif * (y.unif - p.unif)^2)
  se.unif <- sqrt(diag(W.unif %*% V.unif %*% W.unif))

  zvalue <- beta.unif/se.unif
  pvalue <- 2*pnorm(-abs(zvalue))
  summary <- data.frame(coefficients = beta.unif, stdErr = se.unif,
                        Zvalue = zvalue, Pvalue = pvalue)
  row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
  return(summary)
}

AdpOptUWLog <- function(X, y, r0, r, optmethod, data, covariate){
  N <- length(y)
  n1 <- sum(y)
  n0 <- N - sum(y)
  d <- dim(X)[2]

  PI.prop <- rep(1/(2*n1), N)
  PI.prop[y==0] <- 1/(2*n0)
  idx.prop <- sample(1:N, r0, T, PI.prop)
  x.prop <- X[idx.prop,]
  y.prop <- y[idx.prop]
  pinv.prop <- 1/PI.prop[idx.prop]

  data.s1 <- data[idx.prop,]
  design <- svydesign(ids=~1,
                      weights = ~1,
                      data=as.data.frame(data.s1))
  fit.uw.prop <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                          covariate)),
                        design=design,
                        family=quasibinomial)$coefficients
  beta.uw.prop <- fit.uw.prop + c(log(n1/n0), rep(0, d-1))

  P.uw.prop  <- 1 - 1 / (1 + exp(c(X %*% beta.uw.prop)))
  p.uw.prop <- P.uw.prop[idx.prop]
  phi.uw.prop <- p.uw.prop * (1 - p.uw.prop)

  p.propT <- 1 - 1 / (1 + exp(c(x.prop %*% fit.uw.prop)))
  phi.propT <- p.propT * (1 - p.propT)
  ldd.prop <- t(x.prop) %*% (x.prop * phi.propT)
  psi.propT <- t(x.prop) %*% (x.prop * ((y.prop - p.propT)^2))
```

```
  if(optmethod == "A"){
    W.uw.prop <- solve(t(x.prop) %*% (x.prop * phi.uw.prop * pinv.prop))
    PI.uw.opt <- sqrt((y - P.uw.prop)^2 * rowSums((X %*% W.uw.prop)^2))
    PI.uw.opt <- c(PI.uw.opt / sum(PI.uw.opt))
  } else if(optmethod == "L") {
    PI.uw.opt <- sqrt((y - P.uw.prop)^2 * rowSums(X^2))
    PI.uw.opt <- c(PI.uw.opt / sum(PI.uw.opt))
  }
  idx.uw.opt <- sample(1:N, r, T, PI.uw.opt)
  x.uw.opt <- X[idx.uw.opt,]
  y.uw.opt <- y[idx.uw.opt]
  pinv.uw.opt <- 1/PI.uw.opt[idx.uw.opt]

  data.s1 <- data[idx.uw.opt,]
  design <- svydesign(ids=~1,
                      weights = 1,
                      data=as.data.frame(data.s1))
  fit.uw.opt <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                         covariate)),
                       design=design,
                       family=quasibinomial)$coefficients
  beta.uw.opt <- fit.uw.opt + beta.uw.prop

  p.optuw  <- 1 - 1 / (1 + exp(c(x.uw.opt %*% fit.uw.opt)))
  phi.optuw <- p.optuw * (1 - p.optuw)
  ldd.optuw <- t(x.uw.opt) %*% (x.uw.opt * phi.optuw)
  W.optuw <- solve(ldd.prop+ldd.optuw)
  beta.optuwcb <- c(W.optuw %*% (ldd.prop %*% beta.uw.prop +
                                   ldd.optuw %*% beta.uw.opt))

  psi.optuw <- t(x.uw.opt) %*% (x.uw.opt * ((y.uw.opt - p.optuw)^2))
  se.optuw <- sqrt(diag(W.optuw %*% (psi.propT + psi.optuw) %*% W.optuw))

  zvalue <- beta.optuwcb/se.optuw
  pvalue <- 2*pnorm(-abs(zvalue))
  summary <- data.frame(coefficients = beta.optuwcb, stdErr = se.optuw,
                        Zvalue = zvalue, Pvalue = pvalue)
  row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
  return(summary)
}

AdpOptPosLog <- function(X, y, r0, r = 1000, optmethod, data, covariate){
```

```
N <- length(y)
n1 <- sum(y)
n0 <- N - sum(y)
d <- dim(X)[2]

PI.prop <- rep(1/(2*n1), N)
PI.prop[y==0] <- 1/(2*n0)
u.prop <- runif(N)
idx.prop <- which(u.prop < (r0 * PI.prop))
x.prop <- X[idx.prop,]
y.prop <- y[idx.prop]
pinv.prop <- 1/pmin(r0 * PI.prop[idx.prop], 1)

data.s1 <- data[idx.prop,]
design <- svydesign(ids=~1, weights=~pmax(r0 * PI.prop[idx.prop], 1),
                    data=as.data.frame(data.s1))
fit.uw.prop <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                        covariate)),
                      design=design,
                      family=quasibinomial)$coefficients
beta.uw.prop <- fit.uw.prop + c(log(n1/n0), rep(0, d-1))

P.uw.prop  <- 1 - 1 / (1 + exp(c(X %*% beta.uw.prop)))
p.uw.prop <- P.uw.prop[idx.prop]
phi.uw.prop <- p.uw.prop * (1 - p.uw.prop)

p.propT <- 1 - 1 / (1 + exp(c(x.prop %*% fit.uw.prop)))
phi.propT <- p.propT * (1 - p.propT)
ldd.prop <- t(x.prop) %*% (x.prop * phi.propT)
psi.propT <- t(x.prop) %*% (x.prop * ((y.prop - p.propT)^2))

if(optmethod == "A"){
  W.uw.prop <- solve(t(x.prop) %*% (x.prop * phi.uw.prop * pinv.prop))
  PI.uw.opt <- sqrt((y - P.uw.prop)^2 * rowSums((X %*% W.uw.prop)^2))
  PHI.uw.opt <- sum(PI.uw.opt[idx.prop] * pinv.prop)
  PI.uw.opt <- c(PI.uw.opt / PHI.uw.opt)
} else if(optmethod == "L") {
  PI.uw.opt <- sqrt((y - P.uw.prop)^2 * rowSums(X^2))
  PHI.uw.opt <- sum(PI.uw.opt[idx.prop] * pinv.prop)
  PI.uw.opt <- c(PI.uw.opt / PHI.uw.opt)
} else if(optmethod == "LCC"){
  PI.uw.opt <- abs(y - P.uw.prop)/r
}
```

```
u.uw.opt <- runif(N)
idx.uw.opt <- which(u.uw.opt < (r * PI.uw.opt))
x.uw.opt <- X[idx.uw.opt,]
y.uw.opt <- y[idx.uw.opt]
pinv.opt <- 1/pmin(r * PI.uw.opt[idx.uw.opt], 1)

data.s1 <- data[idx.uw.opt,]
design <- svydesign(ids=~1, weights=~pmax(r * PI.uw.opt[idx.uw.opt], 1),
                    data=as.data.frame(data.s1))
fit.uw.opt <- svyglm(as.formula(paste0(colnames(data)[d], "~",
                                       covariate)),
                     design=design,
                     family=quasibinomial)$coefficients

beta.uw.opt <- fit.uw.opt + beta.uw.prop

p.optuw  <- 1 - 1 / (1 + exp(c(x.uw.opt %*% fit.uw.opt)))
phi.optuw <- p.optuw * (1 - p.optuw)
ldd.optuw <- t(x.uw.opt) %*% (x.uw.opt * phi.optuw)

if(optmethod != "LCC"){
  W.optuw <- solve(ldd.prop+ldd.optuw)
  beta.optuwcb <- c(W.optuw %*% (ldd.prop %*% beta.uw.prop +
                                   ldd.optuw %*% beta.uw.opt))

  psi.optuw <- t(x.uw.opt) %*% (x.uw.opt * ((y.uw.opt - p.optuw)^2))
  se.optuw <- sqrt(diag(W.optuw %*% (psi.propT + psi.optuw) %*% W.optuw))
} else {
  beta.optuwcb <- beta.uw.opt
  W.optuw <- solve(ldd.optuw)
  psi.optuw <- t(x.uw.opt) %*% (x.uw.opt * ((y.uw.opt - p.optuw)^2))
  se.optuw <- sqrt(diag(W.optuw %*% (psi.optuw) %*% W.optuw))
}

zvalue <- beta.optuwcb/se.optuw
pvalue <- 2*pnorm(-abs(zvalue))

summary <- data.frame(coefficients = beta.optuwcb, stdErr = se.optuw,
                      Zvalue = zvalue, Pvalue = pvalue)
row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
sample.size <- data.frame(pilot.sample.size = length(y.prop),
                          second.sample.size = length(y.uw.opt))
```

```
  return(list(summary, sample.size))
}

AdpOptSubPoi <- function(X, y, r0, r, optmethod, delta.quant = 0.05){
  N <- length(y)
  d <- dim(X)[2]

  idx.prop <- sample(1:N, r0, T)
  x.prop <- X[idx.prop,]
  y.prop <- y[idx.prop]
  pinv.prop <- rep(N, r0)

  beta.prop <- glm(y.prop ~ x.prop[, -1],
                   family = "quasipoisson")$coefficients
  lbd.prop <- exp(c(X %*% beta.prop))
  W.prop <- solve(t(X) %*% (X * lbd.prop))
  k.prop <- (y - lbd.prop)^2
  k.prop <- pmax(k.prop, quantile(k.prop, delta.quant)^2)

  if(optmethod == "A"){
    PI.opt <- sqrt(k.prop * rowSums((X %*% W.prop)^2))
    PI.opt <- PI.opt/sum(PI.opt)
  } else if(optmethod == "L"){
    PI.opt <- sqrt(k.prop * rowSums(X^2))
    PI.opt <- PI.opt/sum(PI.opt)
  } else if(optmethod == "uniform") {
    PI.opt <- rep(1/N, N)
  }

  idx.opt <- sample(1:N, r, T, PI.opt)
  x.opt <- X[c(idx.prop, idx.opt),]
  y.opt <- y[c(idx.prop, idx.opt)]
  pinv.opt <- c(pinv.prop, 1/PI.opt[idx.opt])

  beta.opt <- glm(y.opt~x.opt[, -1],
                  weights = pinv.opt,
                  family = "quasipoisson")$coefficients

  lbd.opt <- exp(c(x.opt %*% beta.opt))
  W.opt <- solve(t(x.opt) %*% (x.opt * lbd.opt * pinv.opt))
  Vc.opt <- t(x.opt) %*% (x.opt * (((y.opt - lbd.opt) * pinv.opt)^2))
  se.opt <- sqrt(diag(W.opt %*% Vc.opt %*% W.opt))
```

```
  zvalue <- beta.opt/se.opt
  pvalue <- 2*pnorm(-abs(zvalue))
  summary <- data.frame(coefficients = beta.opt, stdErr = se.opt,
                        Zvalue = zvalue, Pvalue = pvalue)
  row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
  return(summary)
}

QuanSub <- function(X, y, r0, r, RR, tau, optmethod = "optL"){
  N <- length(y)
  d <- dim(X)[2]
  beta.RR <- matrix(NA, nrow = d, ncol = RR)

  idx.simp <- sample(1:N, r0, T)
  x.simp <- X[idx.simp,]
  y.simp <- y[idx.simp]
  fit.simp <- rq(y.simp ~ x.simp[, -1], tau=tau)
  beta.simp <- fit.simp$coefficients
  ep <- c(y - X %*% beta.simp)
  Ie.simp  <- (ep < 0)
  PI.mvc <- sqrt((tau - Ie.simp)^2 * rowSums(X^2))
  PI.mvc <- PI.mvc/sum(PI.mvc)

  if (optmethod == "L") {
    for (rr in 1:RR) {
      idx <- sample(1:N, r, T, PI.mvc)
      x.mVc <- X[idx,]
      y.mVc <- y[idx]
      pinv <- 1/PI.mvc[idx]
      fit <- rq(y.mVc ~ x.mVc[, -1], tau=tau, weights=pinv)
      beta.RR[, rr] <- fit$coefficients
      efs <- 1- sum(PI.mvc^2)*((r*RR-1)/2)
    }
  } else if (optmethod == "uniform"){
    for (rr in 1:RR) {
      idx <- sample(1:N, r, T)
      x.unif <- X[idx,]
      y.unif <- y[idx]
      fit <- rq(y.unif ~ x.unif[, -1], tau=tau)
      beta.RR[, rr] <- fit$coefficients
      efs <- 1- ((r*RR-1)/(2*N))
    }
  }
```

```
  beta <- rowMeans(beta.RR)
  se.beta <- sqrt(rowSums((beta.RR - beta)^2)/(RR*(RR-1)*efs))
  zvalue <- beta/se.beta
  pvalue <- 2*pnorm(-abs(zvalue))
  summary <- data.frame(coefficients = beta, stdErr = se.beta,
                        Zvalue = zvalue, Pvalue = pvalue)
  row.names(summary) <- c("intercept", paste0("beta", 1:(d-1)))
  return(summary)
}
```