Web-based Supplementary Materials: Appendix C

1 Introduction

We have included R functions and code for obtaining simulation results reported in Section 3 of the paper.

2 R code

2.1 Loading packages

The R packages used in simulation studies can be downloaded from CRAN (https://cran.r-project.org) and loaded locally.

```
install.packages("glmnet")
library(glmnet)
install.packages("devtools")
devtools::install_github("tdhock/WeightedROC")
library(WeightedROC)
install.packages("bootstrap")
library(bootstrap)
install.packages("ggpubr")
library(ggpubr)
```

2.2 Simulation: Binary matching variable

The function **cohort.binary()** is used to simulate data from a cohort of sample size N, where the matching variable Z is binary.

```
cohort.binary <- function(N, alpha, gamma, beta, delta){
  z <- rbinom(N, 1, prob=0.5)
  x <- c()
  x[z==0] <- rnorm(sum(z==0), 0, 1)
  x[z==1] <- rnorm(sum(z==1), delta, 1)
  linear.pred <- alpha + x*beta + z*gamma
  prob.y <- exp(linear.pred)/(1+exp(linear.pred))
  y=rbinom(N, 1, prob=prob.y)
  cohort.data <- data.frame(y, x, z)
  colnames(cohort.data) <- c("y",  "x" ,"z")
  return(cohort.data)
}
```

Inputs to the function alpha, gamma, beta, delta are as defined in 3.1 corresponding to a continuous matching variable, Z.

- **N**: Number of subjects in the cohort.
- $\boldsymbol{\alpha}$: Intercept of the population logistic model.

- $\boldsymbol{\gamma}$: Coefficient associated with the binary matching variable Z.
- $\boldsymbol{\beta}$: Coefficient associated with the biomarker X.
- $\boldsymbol{\delta}$: represents the mean shift in levels of biomarker X across levels of binary matching variable Z.

The output of the cohort.binary() function is a $N \times 3$ matrix with columns corresponding to the binary response Y, biomarker X and binary matching variable Z.

The function **mcc()** is used to select a matched case control dataset of size n case-control pairs.

```
resample <- function(x, ...) x[sample.int(length(x), ...)]

mcc <- function(cohort.data, n){
  case <- which(cohort.data[, 1]==1)
  control <- which(cohort.data[, 1]==0)
  if(sum(cohort.data[,1]==1)>=n){
    case.id <- resample(case, n, replace=F)
  } else {
    case.id <- case
  }
  control.id <- c()
  non.match <- c()
  i <- 1
  while(i <= length(case.id)){
    strat <- cohort.data[case.id[i], dim(cohort.data)[2]]
    id.match <- control[which(cohort.data[control, dim(cohort.data)[2]]==strat)]
    if(length(id.match) >=1){
      control.id[i] <- resample(id.match, 1, replace=F)
      control <- control[-which(control==control.id[i])]
    } else{
      non.match <- c(non.match, i)
      unsel.case.id <- setdiff(case, case.id)
      if(length(unsel.case.id) > 0){
        newcase <- resample(unsel.case.id, 1, replace=F)
        case.id <- c(case.id, newcase)
      }else{
        non.match <- c(non.match, i)
      }
    }
    i <- i + 1
  }
  if(length(non.match) > 0){
    case.id <- case.id[-non.match]
    control.id <- control.id[-non.match]
  }
  data <- rbind(cohort.data[case.id, ], cohort.data[control.id, ])
  strata <- rep(1:length(case.id), 2)
  mcc.data <- data.frame(data, strata)
  return(mcc.data)
}
```

Inputs to the *mcc()* function:

- **cohort.data**: Data matrix of size $N \times 3$ generated from *cohort()* function. The columns of the input dataset include the binary response Y, biomarker X, binary, matching variable Z.

- **n**: Number of matched **pairs** of cases and controls, where the matching is on variable Z.

The mcc() function outputs a dataset of size $2n \times 4$, where the columns correspond to binary response Y, biomarker X, binary matching variable Z and a matched pair indicator.

The function **weight.calc.binary()** obtains the sampling weights from the cohort data, for the setting in which the matching variable is binary.

```
weight.calc.binary <- function(cohort.data, mcc.data){
  nC <- c() #cohort case
  nC0 <- c() #cohort control
  mC <- c() #MCC case
  mC0 <- c() #MCC control
  sort.w <- sort(unique(cohort.data$w))

  for(i in 1:length(sort.w)){
    nC0[i] <- dim(cohort.data[cohort.data$w==sort.w[i] & cohort.data$y == 0, ])[1]
    nC[i] <- dim(cohort.data[cohort.data$w==sort.w[i] & cohort.data$y == 1, ])[1]
    mC0[i] <- dim(mcc.data[mcc.data$w==sort.w[i] & mcc.data$y == 0, ])[1]
    mC[i] <- dim(mcc.data[mcc.data$w==sort.w[i] & mcc.data$y == 1, ])[1]
  }
  weight <- rep(1, dim(mcc.data)[1])
  for (i in 1:length(sort.w)){
    weight[(mcc.data$w==sort.w[i] & mcc.data$y==1)] <- nC[i]/mC[i]
    weight[(mcc.data$w==sort.w[i] & mcc.data$y==0)] <- nC0[i]/mC0[i]
  }
  return(weight)
}
```

Inputs to the *weight.calc.binary()* function incude:

- **cohort.data**: $N \times 3$ cohort dataset generated from *cohort()* function.
- **mcc.data**: $2n \times 4$ matched case-control data generated from *mcc()* function.

Output from the *weight.calc()* function is a vector of weights for n matched pairs in ascending order of the matched pair indicator (Column 4 of the output from mcc()).

2.3 Simulation: Continous matching variable

The function **cohort.continuous()** is used to simulate data from a cohort of sample size N, where the matching variable Z is continuous.

```
cohort.continuous <- function(N, alpha, gamma, beta, rho){
  z <- rnorm(N, 1, 1.5)
  w <- as.integer(cut(z, quantile(z, probs=0:10/10), include.lowest=TRUE))
  x <- rnorm(N, mean=rho*w, sd=sqrt(1-rho^2))
  linear.pred <- alpha + x*beta + z*gamma
  prob.y <- exp(linear.pred)/(1+exp(linear.pred))
  y=rbinom(N, 1, prob=prob.y)
  cohort.data <- data.frame(y, x, z, w)
  colnames(cohort.data) <- c("y",  "x" ,"z", "w")
  return(cohort.data)
}
```

Inputs to the function cohort.continuous(), namely alpha, gamma, beta, rho are as defined in 3.1 corresponding to a continuous matching variable, Z.

- **N**: Number of subjects in the cohort.
- $\boldsymbol{\alpha}$: Intercept of the population logistic model.
- $\boldsymbol{\gamma}$: Coefficient associated with the binary matching variable Z.
- $\boldsymbol{\beta}$: Coefficient associated with the biomarker X.
- $\boldsymbol{\rho}$: represents the represents the correlation between biomarker X and continous matching variable Z.

The output of the cohort.continuous() function is a $N \times 4$ matrix with columns corresponding to the binary response Y, biomarker X, continuous variable Z and categorical matching variable W.

The function **weight.calc.continuous()** obtains the sampling weights from the cohort data, for the setting in which the matching variable is continuous.

```
weight.calc.continuous <- function(cohort.data, mcc.data){
  nC <- c() #cohort case
  nC0 <- c() #cohort control
  mC <- c() #MCC case
  mC0 <- c() #MCC control
  sort.w <- sort(unique(cohort.data$w))

  for(i in 1:length(sort.w)){
    nC0[i] <- dim(cohort.data[cohort.data$w==sort.w[i] & cohort.data$y == 0, ])[1]
    nC[i] <- dim(cohort.data[cohort.data$w==sort.w[i] & cohort.data$y == 1, ])[1]
    mC0[i] <- dim(mcc.data[mcc.data$w==sort.w[i] & mcc.data$y == 0, ])[1]
    mC[i] <- dim(mcc.data[mcc.data$w==sort.w[i] & mcc.data$y == 1, ])[1]
  }
  #  J <- nC0/nC
  weight <- rep(1, dim(mcc.data)[1])
  #  weight[mcc.data$y==1] <- 1
  for (i in 1:length(sort.w)){
    weight[(mcc.data$w==sort.w[i] & mcc.data$y==1)] <- nC[i]/mC[i]
    weight[(mcc.data$w==sort.w[i] & mcc.data$y==0)] <- nC0[i]/mC0[i]
  }
  return(weight)
}
```

Inputs to the *weight.calc.continuous()* function incude:

- **cohort.data**: $N \times 4$ cohort dataset generated from *cohort()* function.
- **mcc.data**: $2n \times 4$ matched case-control data generated from *mcc()* function.

Output from the *weight.calc()* function is a vector of weights for n matched pairs in ascending order of the matched pair indicator (Column 4 of the output from mcc()).

Examples

```
# Binary Matching Variable
cohort.data <- cohort.binary(N=1000, alpha=-2.1, gamma=0, beta=0.5, delta=0)
mcc.data <- mcc(cohort.data, n=100)
weight <- weight.calc.binary(cohort.data, mcc.data)

# Continuous Matching Variable
cohort.data <- cohort.continuous(N=1000, alpha=-2.1, gamma=0, beta=0.5, rho=0)
```

```
mcc.data <- mcc(cohort.data, n=100)
weight <- weight.calc.continuous(cohort.data, mcc.data)
```

2.4 ROC curve and AUC estimation

The function AUC.Calc() outputs the weighted and unweighted AUC estimates from the following approaches using the matched case control dataset (mcc.data): (1) Logistic regression (LR); (2) Conditional logistic regression (CLR); (3) Two-stage procedure; (4) Simultaneous procedure. Both weighted and unweighted estimates from the above mentioned four methods are compared to AUC estimated from a logistic regression model fit to the entire cohort (gold standard).

```
AUC.Calc <- function(cohort.data, mcc.data, bootnum){
  library(survival)
  if(length(unique(cohort.data$z))==2){
    weights <- weight.calc.binary(cohort.data, mcc.data)
  } else{
    weights <- weight.calc.continuous(cohort.data, mcc.data)
  }
  ############# Logistic regression on cohort dataset (gold standard) #############
  cohort.ucl <- glm(y ~ x + z, data=cohort.data, family="binomial")
  cohort.auc <- WeightedAUC(WeightedROC(predict(cohort.ucl), cohort.data$y,
                                        rep(1, dim(cohort.data)[1])))

  ############# Conditional Logistic Regression  #############
  mcc.clr <- clogit(y ~ x + strata(strata), data=mcc.data,  method="approximate")
  clr.est.beta  <- coef(mcc.clr)
  pred.clr.train <- clr.est.beta * mcc.data$x
  clr.train.auc <- WeightedAUC(WeightedROC(pred.clr.train, mcc.data$y,
                                           rep(1, dim(mcc.data)[1])))
  w.clr.train.auc <- WeightedAUC(WeightedROC(pred.clr.train, mcc.data$y,
                                             weights))

  ############# Logistic Regression #############
  mcc.uclr <- glm(y ~ x + z, data=mcc.data, family="binomial")
  uclr.est.beta <- coef(mcc.uclr)[2]
  uclr.est.gamma <- coef(mcc.uclr)[3]
  uclr.train.auc <- WeightedAUC(WeightedROC(predict(mcc.uclr), mcc.data$y,
                                            rep(1, dim(mcc.data)[1])))
  w.uclr.train.auc <- WeightedAUC(WeightedROC(predict(mcc.uclr), mcc.data$y, weights))

  ############# Two-stage algorithm #############
  fitz <- glm(y ~ z, data=cohort.data, family="binomial")
  fz <- -predict(fitz, newdata=mcc.data)
  data1 <- cbind(mcc.data, fz)
  fit.multi <- glm(y ~ z, offset=fz + as.matrix(data1$x)*coef(mcc.clr), data=data1,
                   family="binomial")
  coef1 <- coef(fit.multi)
  multi.est.gamma <- coef1[-1]
  multi.train.p.score <- rep(coef1[1], dim(mcc.data)[1]) +
    as.numeric(mcc.data[,2])*coef(mcc.clr) + as.matrix(mcc.data[,3])*multi.est.gamma
  multi.train.auc <- WeightedAUC(WeightedROC(multi.train.p.score , mcc.data$y,
                                             rep(1,dim(mcc.data)[1])))
  w.multi.train.auc <- WeightedAUC(WeightedROC(multi.train.p.score , mcc.data$y, weights))
```

```
############## Simultaneous algorithm #############
fit.simul <- glm(y ~ x+z , offset=fz, data=data1, family="binomial")
coef2 <- coef(fit.simul)
simul.est.beta <- coef2[2]
simul.est.gamma <- coef2[3]
simul.train.p.score <- rep(coef2[1], dim(mcc.data)[1]) +
  as.numeric(mcc.data[,2])*coef2[2] + as.matrix(mcc.data[,3])*simul.est.gamma
simul.train.auc <- WeightedAUC(WeightedROC(simul.train.p.score, mcc.data$y,
                                           rep(1, dim(mcc.data)[1])))
w.simul.train.auc <- WeightedAUC(WeightedROC(simul.train.p.score, mcc.data$y, weights))

theta22 <- function(vec, mcc.data){
  bootdata <- rbind(mcc.data[vec, ], mcc.data[vec+dim(mcc.data)[1]/2, ])
  boot.weight <- c(weights[vec], weights[vec+dim(mcc.data)[1]/2])
  mcc.clr <- clogit(y ~ x + strata(strata), data=bootdata,  method="approximate")
  pred.clr.boot <- coef(mcc.clr) * bootdata$x
  w.clr.boot.auc <- WeightedAUC(WeightedROC(pred.clr.boot, bootdata$y, boot.weight))
  return(w.clr.boot.auc)
}

theta12 <- function(vec, mcc.data){
  bootdata <- rbind(mcc.data[vec, ], mcc.data[vec+dim(mcc.data)[1]/2, ])
  boot.weight <- c(weights[vec], weights[vec+dim(mcc.data)[1]/2])
  mcc.uclr <- glm(y ~ x + z, data=bootdata, family="binomial")
  w.uclr.boot.auc <- WeightedAUC(WeightedROC(predict(mcc.uclr), bootdata$y, boot.weight))
  return(w.uclr.boot.auc)
}

theta32 <- function(vec, mcc.data){
  bootdata <- rbind(mcc.data[vec, ], mcc.data[vec+dim(mcc.data)[1]/2, ])
  boot.weight <- c(weights[vec], weights[vec+dim(mcc.data)[1]/2])
  mcc.clr <- clogit(y ~ x + strata(strata), data=bootdata,  method="approximate")

  fz <- -predict(fitz, newdata=bootdata)
  data1 <- cbind(bootdata, fz)
  fit.multi.boot <- glm(y ~ z, offset=fz + as.matrix(data1$x)*coef(mcc.clr),
                        data=data1, family="binomial")
  coef1 <- coef(fit.multi.boot)
  multi.boot.p.score <- rep(coef1[1], dim(bootdata)[1]) +
    as.numeric(bootdata$x)*coef(mcc.clr) + as.matrix(bootdata$z)*coef1[2]
  w.multi.boot.auc <- WeightedAUC(WeightedROC(multi.boot.p.score, bootdata$y, boot.weight))
  return(w.multi.boot.auc)
}
############## simultaneous  #############

theta42 <- function(vec, mcc.data){
  bootdata <- rbind(mcc.data[vec, ], mcc.data[vec+dim(mcc.data)[1]/2, ])
  boot.weight <- c(weights[vec], weights[vec+dim(mcc.data)[1]/2])
  fz <- -predict(fitz, newdata=bootdata)
  data1 <- cbind(bootdata, fz)
  fit.simul.boot <- glm(y ~ x+z , offset=fz, data=data1, family="binomial")
  coef2 <- coef(fit.simul.boot)
  simul.boot.p.score <- rep(coef2[1], dim(bootdata)[1]) +
```

```
      as.numeric(bootdata[,2])*coef2[2]  + as.matrix(bootdata[,3])*coef2[3]
    w.simul.boot.auc <- WeightedAUC(WeightedROC(simul.boot.p.score, bootdata$y, boot.weight))
    return(w.simul.boot.auc)
  }

  results1 <- bcanon(1:(dim(mcc.data)[1]/2), bootnum, theta12, mcc.data)$confpoints[c(1,8),2]
  results2 <- bcanon(1:(dim(mcc.data)[1]/2), bootnum, theta22, mcc.data)$confpoints[c(1,8),2]
  results3 <- bcanon(1:(dim(mcc.data)[1]/2), bootnum, theta32, mcc.data)$confpoints[c(1,8),2]
  results4 <- bcanon(1:(dim(mcc.data)[1]/2), bootnum, theta42, mcc.data)$confpoints[c(1,8),2]

  cov.uclr <- ifelse (cohort.auc>=results1[1] & cohort.auc <= results1[2], 1, 0)
  cov.clr <- ifelse (cohort.auc>=results2[1] & cohort.auc <= results2[2], 1, 0)
  cov.multi <- ifelse (cohort.auc>=results3[1] & cohort.auc <= results3[2], 1, 0)
  cov.simul <- ifelse (cohort.auc>=results4[1] & cohort.auc <= results4[2], 1, 0)


  if(length(unique(cohort.data$z))==2){
    uclr <- c(uclr.est.beta, uclr.est.gamma, cohort.auc,
              uclr.train.auc, w.uclr.train.auc, cov.uclr)
    clr <- c(clr.est.beta, 0, cohort.auc,
             clr.train.auc, w.clr.train.auc, cov.clr)
    multi <- c(clr.est.beta, multi.est.gamma, cohort.auc,
               multi.train.auc, w.multi.train.auc, cov.multi)
    simul <- c(simul.est.beta, simul.est.gamma, cohort.auc,
               simul.train.auc, w.simul.train.auc, cov.simul)
  } else {
    uclr <- c(uclr.est.beta, uclr.est.gamma, cohort.auc,
              uclr.train.auc, w.uclr.train.auc,cov.uclr)
    clr <- c(clr.est.beta, 0, cohort.auc,
             clr.train.auc, w.clr.train.auc,cov.clr)
    multi <- c(clr.est.beta, multi.est.gamma, cohort.auc,
               multi.train.auc, w.multi.train.auc,cov.multi)
    simul <- c(simul.est.beta, simul.est.gamma, cohort.auc,
               simul.train.auc, w.simul.train.auc,cov.simul)
  }
  result <- rbind(uclr, clr, multi, simul)
  colnames(result) <- c("beta Estimate", "gamma Estimate", "True AUC",
                        "Unweighted AUC Estimate", "Weighted AUC Estimate","CovBCa")
  return(result)
}
```

Inputs to the *AUC.Calc()* function include:

- **cohort.data**: A N $\times$ 3 matrix output by the *cohort.binary()* function or an $N \times 4$ matrix output by the *cohort.continuous()* function.
- **mcc.data**: A 2n $\times$ 4 matrix output by *mcc()* function.
- **bootnum**: Number of bootstrapped samples used for estimating Bca confidence intervals.

Output of the *AUC.Calc()* function:
The output includes the following: For each of the four methods (namely, (1) Logistic regression (LR); (2) Conditional logistic regression (CLR); (3) Two-stage algorithm (multi); (4) Simultaneous algorithm (simul)), estimates of $\beta(Column1), \gamma$ (Column 2), unweighted and weighted AUC (Columns 4-5). Also, included for comparison is the true AUC from the simulated cohort (Column 3).

Example

```
library(survival)
cohort.data <- cohort.binary(N=1000, alpha=-2.1, gamma=0, beta=0.5, delta=0)
mcc.data <- mcc(cohort.data, n=100)
result <- AUC.Calc(cohort.data, mcc.data, bootnum=100)
```

3 Tables

Results reported in Tables 1-3 (Main paper) and Web Tables 1-3 were generated using the code below:

```
create.table <- function(N=N, n=n, alpha=alpha, gamma=gamma, beta=beta, delta=delta,
                         bootnum=bootnum,binary=1){
  r<- list()
  if(binary==1){
      cohort.train <- cohort.binary(N, alpha=alpha, gamma=gamma, beta=beta, delta=delta)
      mcc.train <- mcc(cohort.train, n)
      weight.train <- weight.calc.binary(cohort.train, mcc.train)
      r <- AUC.Calc2(cohort.train, mcc.train, bootnum=bootnum)
    }else {
      cohort.train <- cohort.continuous(N, alpha=alpha, gamma=gamma, beta=beta, rho=delta)
      mcc.train <- mcc(cohort.train, n)
      weight.train <- weight.calc.continuous(cohort.train, mcc.train)
      r<- AUC.Calc2(cohort.train, mcc.train, bootnum=bootnum)
    }
  return(r)
}

runsim <- function(sim, N, n, alpha, gamma, beta, delta, bootnum,binary=1, numcores){
  result <- mclapply(1:sim, function(x) create.table(N, n, alpha, gamma,
                                                     beta, delta, bootnum,binary),
                     mc.cores=numcores)
  unweighted.auc.bias <- lapply(result, function(x) x[, 4]-x[ ,3])
  weighted.auc.bias <- lapply(result, function(x) x[, 5]-x[ ,3])

  beta.bias <- lapply(result, function(x) x[, 1]- beta)
  gamma.bias <- lapply(result, function(x) x[, 2] - gamma)
  result1 <- lapply(1:sim, function(x) cbind(result[[x]], unweighted.auc.bias[[x]],
                                             weighted.auc.bias[[x]], beta.bias[[x]],
                                             gamma.bias[[x]]))
  re1 <- colMeans(t(sapply(result1, function(x) x[1,])))
  re2 <- colMeans(t(sapply(result1, function(x) x[2,])))
  re3 <- colMeans(t(sapply(result1, function(x) x[3,])))
  re4 <- colMeans(t(sapply(result1, function(x) x[4,])))
  re <- rbind(re1, re2, re3, re4)
  se1 <- apply(sapply(result1, function(x) x[1, ])[c(4:5, 7:10), ], 1, function(x) sd(x)/sqrt(sim))
  se2 <- apply(sapply(result1, function(x) x[2, ])[c(4:5, 7:10), ], 1, function(x) sd(x)/sqrt(sim))
  se3 <- apply(sapply(result1, function(x) x[3, ])[c(4:5, 7:10), ], 1, function(x) sd(x)/sqrt(sim))
  se4 <- apply(sapply(result1, function(x) x[4, ])[c(4:5, 7:10), ], 1, function(x) sd(x)/sqrt(sim))
  ses <- rbind(se1, se2, se3, se4)
  lm11 <- quantile(sapply(result1, function(x) x[1, ])[4, ], c(0.025, 0.975))
  lm22 <- quantile(sapply(result1, function(x) x[2, ])[4, ], c(0.025, 0.975))
  lm33 <- quantile(sapply(result1, function(x) x[3, ])[4, ], c(0.025, 0.975))
```

```
  lm44 <- quantile(sapply(result1, function(x) x[4, ])[4, ], c(0.025, 0.975))
  lms2 <- rbind(lm11, lm22, lm33, lm44)
  lm1 <- quantile(sapply(result1, function(x) x[1, ])[5, ], c(0.025, 0.975))
  lm2 <- quantile(sapply(result1, function(x) x[2, ])[5, ], c(0.025, 0.975))
  lm3 <- quantile(sapply(result1, function(x) x[3, ])[5, ], c(0.025, 0.975))
  lm4 <- quantile(sapply(result1, function(x) x[4, ])[5, ], c(0.025, 0.975))
  lms <- rbind(lm1, lm2, lm3, lm4)
  colnames(re) <- c("beta estimate", "gamma estimate", "True AUC", "Unweighted AUC",
                    "Weighted AUC", "CovBCa", "unweighted.auc.bias", "Weighted.auc.bias",
                    "beta.bias", "gamma.bias")
  colnames(ses) <- c("SE of Unweighted AUC estimates", "SE of weighted AUC estimates",
                     "SE.UnweightedAUC.bias", "SE.WeightedAUC.bias",
                     "SE.beta.bias", "SE.gamma.bias")
  colnames(lms) <- c("2.5th", "97.5th")
  colnames(lms2) <- c("2.5th", "97.5th")

  param <- matrix(rep(c(beta, gamma, delta),4), nrow=4, byrow=T)
  colnames(param) <- c("beta", "gamma", "delta")
  result2 <- cbind(param, re, ses, lms2, lms)
  rownames(result2) <- c("LR", "CLR", "Two-stage", "Simultaneous")
  return(list(result1=result1, result2=result2))
}
# Table 1
param11 <- runsim(sim=500, N=2000, n=200, alpha=-1.9, gamma=0, beta=0.5, delta=0,
                  bootnum=2000,binary=1, numcores=numcores)
save(param11, file="param11.rda")
param12 <- runsim(sim=500, N=2000, n=200, alpha=-2, gamma=0.25, beta=0.5, delta=0,
                  bootnum=2000,binary=1, numcores=numcores)
save(param12, file="param12.rda")
param13 <- runsim(sim=500, N=2000, n=200, alpha=-2.2, gamma=0.5, beta=0.5, delta=0,
                  bootnum=2000,binary=1, numcores=numcores)
save(param13, file="param13.rda")
param14 <- runsim(sim=500, N=2000, n=200, alpha=-2.3, gamma=0.75, beta=0.5, delta=0,
                  bootnum=2000,binary=1, numcores=numcores)
save(param14, file="param14.rda")
table1 <- rbind(param11[[2]], param12[[2]], param13[[2]], param14[[2]])

# Table 2
param21 <- runsim(sim=500, N=2000, n=200, alpha=-2, gamma=0, beta=0.5, delta=0.5,
                  bootnum=2000,binary=1, numcores=numcores)
save(param21, file="param21.rda")
param22 <- runsim(sim=500, N=2000, n=200, alpha=-2.2, gamma=0.25, beta=0.5, delta=0.5,
                  bootnum=2000,binary=1, numcores=numcores)
save(param22, file="param22.rda")
param23 <- runsim(sim=500, N=2000, n=200, alpha=-2.3, gamma=0.5, beta=0.5, delta=0.5,
                  bootnum=2000,binary=1, numcores=numcores)
save(param23, file="param23.rda")
param24 <- runsim(sim=500, N=2000, n=200, alpha=-2.5, gamma=0.75, beta=0.5, delta=0.5,
                  bootnum=2000,binary=1, numcores=numcores)
save(param24, file="param24.rda")
table2 <- rbind(param21[[2]], param22[[2]], param23[[2]], param24[[2]])

# Table 3
```

```
param31 <- runsim(sim=500, N=2000, n=200, alpha=-2.2, gamma=0, beta=0.5, delta=1,
                  bootnum=2000,binary=1, numcores=numcores)
save(param31, file="param31.rda")
param32 <- runsim(sim=500, N=2000, n=200, alpha=-2.4, gamma=0.25, beta=0.5, delta=1,
                  bootnum=2000,binary=1, numcores=numcores)
save(param32, file="param32.rda")
param33 <- runsim(sim=500, N=2000, n=200, alpha=-2.6, gamma=0.5, beta=0.5, delta=1,
                  bootnum=2000,binary=1, numcores=numcores)
save(param33, file="param33.rda")
param34 <- runsim(sim=500, N=2000, n=200, alpha=-2.7, gamma=0.75, beta=0.5, delta=1,
                  bootnum=2000,binary=1, numcores=numcores)
save(param34, file="param34.rda")
table3 <- rbind(param31[[2]], param32[[2]], param33[[2]], param34[[2]])

# Web Table 1
param41 <- runsim(sim=500, N=2000, n=200, alpha=-2, gamma=0, beta=0.5, delta=0,
                  bootnum=2000,binary=2, numcores=numcores)
save(param41, file="param41.rda")
param42 <- runsim(sim=500, N=2000, n=200, alpha=-2.3, gamma=0.25, beta=0.5, delta=0,
                  bootnum=2000, binary=2, numcores=numcores)
save(param42, file="param42.rda")
param43 <- runsim(sim=500, N=2000, n=200, alpha=-2.7, gamma=0.5, beta=0.5, delta=0,
                  bootnum=2000, binary=2, numcores=numcores)
save(param43, file="param43.rda")
param44 <- runsim(sim=500, N=2000, n=200, alpha=-3.1, gamma=0.75, beta=0.5, delta=0,
                  bootnum=2000, binary=2, numcores=numcores)
save(param44, file="param44.rda")
table4 <- rbind(param41[[2]], param42[[2]], param43[[2]], param44[[2]])

# Web Table 2
param51 <- runsim(sim=500, N=2000, n=200, alpha=-2.9, gamma=0, beta=0.5, delta=0.35,
                  bootnum=2000, binary=2, numcores=numcores)
save(param51, file="param51.rda")
param52 <- runsim(sim=500, N=2000, n=200, alpha=-3.3, gamma=0.25, beta=0.5, delta=0.35,
                  bootnum=2000, binary=2, numcores=numcores)
save(param52, file="param52.rda")
param53 <- runsim(sim=500, N=2000, n=200, alpha=-3.8, gamma=0.5, beta=0.5, delta=0.35,
                  bootnum=2000, binary=2, numcores=numcores)
save(param53, file="param53.rda")
param54 <- runsim(sim=500, N=2000, n=200, alpha=-4.3, gamma=0.75, beta=0.5, delta=0.35,
                  bootnum=2000, binary=2, numcores=numcores)
save(param54, file="param54.rda")
table5 <- rbind(param51[[2]], param52[[2]], param53[[2]], param54[[2]])

# Web Table 3
param61 <- runsim(sim=500, N=2000, n=200, alpha=-3.1, gamma=0, beta=0.5, delta=0.5,
                  bootnum=2000, binary=2, numcores=numcores)
save(param61, file="param61.rda")
param62 <- runsim(sim=500, N=2000, n=200, alpha=-3.4, gamma=0.25, beta=0.5, delta=0.5,
                  bootnum=2000, binary=2, numcores=numcores)
save(param62, file="param62.rda")
start = proc.time()
param63 <- runsim(sim=500, N=2000, n=200, alpha=-3.8, gamma=0.5, beta=0.5, delta=0.5,
```

```
                  bootnum=2000, binary=2, numcores=numcores)
proc.time() - start
save(param63, file="param63.rda")
param64 <- runsim(sim=500, N=2000, n=200, alpha=-4.2, gamma=0.75, beta=0.5, delta=0.5,
                  bootnum=200, binary=2, numcores=numcores)
save(param64, file="param64.rda")
table6 <- rbind(param61[[2]], param62[[2]], param63[[2]], param64[[2]])
```

4 Graphs

Results reported in Figures 1-3 (Main paper) and Web Figures 1-3 were generated using the code below:

```
library(ggplot2)
library(ggthemes)
library(ggpubr)
re <- data.frame(names = rownames(result), result)
dat1 <- re %>%
  gather(method, AUC, Unweighted.AUC:Weighted.AUC)
dat2 <- re %>%
  gather(method, AUC.bias, unweighted.auc.bias:Weighted.auc.bias)
dat3 <- re %>%
  gather(method, SE.AUC, SE.of.Unweighted.AUC.estimates:SE.of.weighted.AUC.estimates)
dat4 <- re %>%
  gather(method, SE.AUC.bias, SE.UnweightedAUC.bias, SE.WeightedAUC.bias.)
dat <- data.frame(cbind(dat1[, c(1:8,11:12, 17:22)], dat2$AUC.bias, dat3$SE.AUC, dat4$SE.AUC.bias))
dat[, 15] <- gsub(".AUC", "", dat[,15])
mymethod <- paste(dat[,1], "-", dat[,15], sep="")
mydat <- dat
mydat <- data.frame(mydat, mymethod)

mymethod2 <- mymethod
mymethod2[mymethod2 == "LR-Weighted"] <- "1a. LR-Weighted"
mymethod2[mymethod2 == "LR-Unweighted"] <- "1b. LR-Unweighted"
mymethod2[mymethod2 == "CLR-Weighted"] <- "2a. CLR-Weighted"
mymethod2[mymethod2 == "CLR-Unweighted"] <- "2b. CLR-Unweighted"
mymethod2[mymethod2 == "Two-stage-Weighted"] <- "3a. Two stage-Weighted"
mymethod2[mymethod2 == "Two-stage-Unweighted"] <- "3b. Two stage-Unweighted"
mymethod2[mymethod2 == "Simultaneous-Weighted"] <- "4a. Simultaneous-Weighted"
mymethod2[mymethod2 == "Simultaneous-Unweighted"] <- "4b. Simultaneous-Unweighted"
mydat1 <- data.frame(mydat, mymethod2)
upper <- mydat1$dat2.AUC.bias + 1.96*mydat1$dat4.SE.AUC.bias
lower <-  mydat1$dat2.AUC.bias - 1.96*mydat1$dat4.SE.AUC.bias
mydat1 <- cbind(mydat1, upper, lower)

p <- ggplot(mydat1, aes(gamma, dat2.AUC.bias, color=factor(delta))) + geom_point() +
  scale_fill_manual(values=c("#00AFBB", "#E7B800", "#FC4E07"))+
  geom_errorbar(aes(x=gamma, ymin= lower, ymax=upper), width=0.10) +
  xlab(expression(gamma)) + ylab(expression("Bias in estimate of AUC")) +
  ggtitle("AUC \n n=200 matched pairs")
facet(p + theme_bw(), facet.by = "mymethod2",
      short.panel.labs = TRUE, # Allow long labels in panels
      ncol=2,
```

```
      panel.labs.background = list(fill = "#00AFBB", color = "#00AFBB")) +
  theme(plot.title = element_text(hjust = 0.5))

####Figure 1 and Web Figure 1

mydat1 <- mydat[(mydat$method == "Weighted"),]
upper <- mydat1$beta.bias + 1.96*mydat1$SE.beta.bias
lower <-  mydat1$beta.bias - 1.96*mydat1$SE.beta.bias
mydat1 <- cbind(mydat1, upper, lower)
mymethod3 <- as.character(mydat1$mymethod2)
mymethod3[mymethod3 == "1a. LR-Weighted"] <- "1. LR"
mymethod3[mymethod3 == "2a. CLR-Weighted"] <- "2. CLR"
mymethod3[mymethod3 == "3a. Two stage-Weighted"] <- "3. Two-stage"
mymethod3[mymethod3 == "4a. Simultaneous-Weighted"] <- "4. Simultaneous"
mydat1 <- cbind(mydat1, mymethod3)

p <- ggplot(mydat1, aes(gamma, beta.bias, color=factor(delta))) + geom_point() +
  ggtitle(expression(paste("Estimating ", beta, " [n=200 matched pairs]")))+
  scale_fill_manual(values=c("#00AFBB", "#E7B800", "#FC4E07"))+
  geom_errorbar(aes(x=gamma, ymin= lower, ymax=upper), width=0.10) +
  xlab(expression(gamma)) + ylab(expression(paste("Bias in estimate of ", beta, sep=""))) +
  ylim(c(-0.05,0.05))
facet(p + theme_bw(), facet.by = "mymethod3",
      short.panel.labs = TRUE, # Allow long labels in panels
      ncol=4,
      panel.labs.background = list(fill = "#00AFBB", color = "#00AFBB")
) + theme(plot.title = element_text(hjust = 0.5))

########Figure 2 and Web Figure2

mydat1 <- mydat[(mydat$method == "Weighted"),]
upper <- mydat1$gamma.bias + 1.96*mydat1$SE.gamma.bias
lower <-  mydat1$gamma.bias - 1.96*mydat1$SE.gamma.bias
mydat1 <- cbind(mydat1, upper, lower)
mymethod3 <- as.character(mydat1$mymethod2)
mymethod3[mymethod3 == "1a. LR-Weighted"] <- "1. LR"
mymethod3[mymethod3 == "2a. CLR-Weighted"] <- "2. CLR"
mymethod3[mymethod3 == "3a. Two stage-Weighted"] <- "3. Two-stage"
mymethod3[mymethod3 == "4a. Simultaneous-Weighted"] <- "4. Simultaneous"
mydat1 <- cbind(mydat1, mymethod3)

p <- ggplot(mydat1, aes(gamma, gamma.bias, color=factor(delta))) + geom_point() +
  ggtitle(expression(paste("Estimating ", gamma, " [n=200 matched pairs]")))+
  scale_fill_manual(values=c("#00AFBB", "#E7B800", "#FC4E07"))+
  geom_errorbar(aes(x=gamma, ymin= lower, ymax=upper), width=0.10) + xlab(expression(gamma)) +
  xlab(expression(gamma)) + ylab(expression(paste("Bias in estimate of ", gamma, sep="")))
facet(p + theme_bw(), facet.by = "mymethod3",
      short.panel.labs = TRUE, # Allow long labels in panels
      ncol=4,
      panel.labs.background = list(fill = "#00AFBB", color = "#00AFBB")
) + theme(plot.title = element_text(hjust = 0.5))

####Fig 3 and Web Figure 3
```