Supplement 1: Simulations — Inverse Probability Weights for Quasi-Continuous Ordinal Exposures with a Binary Outcome: Method Comparison and Case Study

Sack, Daniel E, Shepherd, Bryan E, Audet, Carolyn M, De Schact, Caroline, Samuels, Lauren R

Updated 2022-07-24 for American Journal of Epidemiology

Setup

Generate Data

Generating data per stipulations of Naimi, Moodie, Auger, Kaufman, Epidemiology 2014; 25: 292-299 (1).

To generate the skewed version of `mage`, which we define as `mage_g`, we started with a gamma distribution with shape equal to `0.5` and scale equal to `500`. We then shifted the mean to `0` and changed it from right skewed to left skew (to make it a more appropriate skew for maternal age). We then normalized the standard deviation to `1` so that we could stretch the distribution to have the same standard deviation as `mage`. Finally, we set the mean of `mage_g` to the mean of `mage` and reallocated any `mage_g` values of less than `11` to the mean age. To generate the updated μ, μ_2, we increased the correlation between `mage_g` and μ_2 from `0.025` to `0.25`.

```
# function to generate data per Naimi et al. specifications,
# but make the exposure ordinal instead of continuous via rounding
sim_data <- function(n) {
  # draw maternal age from normal distribution
  mage <- rnorm(n, 29.84, sqrt(21.60))

  # maternal age from gamma distribution for a conditionally normal
  # but marginally not normal covariate
  m1 <- rgamma(n, shape = 0.5, scale = 5000) # make very skewed distribution, sims is 5 and 5
  m2 <- (m1 - mean(m1)) * -1 # shift the mean to zero and
  # flip the direction of the skew (so left instead of right skewed)
  m3 <- m2 / sd(m2) # makes sd 1
  m4 <- m3 * sd(mage) # stretch so it has the sd of mage
  mage_g <- m4 + mean(mage) # make it have the same mean as mage
  mage_g[mage_g < 11] <- mean(mage_g) # make any values < 0 the mean,
  # since maternal age cannot really be under 11

  # draw paternal age from normal distribution
  page <- rnorm(n, 32.52, sqrt(30.45))

  # establish parity with same parameters as Naimi et al.
  parityA <- runif(n)
  parity <- ifelse(parityA <= 0.24, 2,
              ifelse(parityA <= 0.24 + 0.07, 3,
```

```
                      ifelse(parityA <= 0.24 + 0.07 + 0.02, 4,
                             ifelse(parityA <= 0.24 + 0.07 + 0.02 + 0.02, 5, 1))))
parity2 <- ifelse(parity == 2, 1, 0)
parity3 <- ifelse(parity == 3, 1, 0)
parity4 <- ifelse(parity == 4, 1, 0)
parity5 <- ifelse(parity == 5, 1, 0)

# mu w/o strong correlation with maternal age
mu_un <- (0.025 * mage) + (0.0025 * page) + (0.00125 * mage * page) -
(0.21 * parity2) - (0.22 * parity3) - (0.45 * parity4) - (0.45 * parity5)

# mu w/ gamma distributed maternal age and strong correlation
mu_g <- (0.25 * mage_g) + (0.0025 * page) + (0.00125 * mage_g * page) -
(0.21 * parity2) - (0.22 * parity3) - (0.45 * parity4) - (0.45 * parity5)

# normal exposure distribution, but round so it's ordinal
x1 <- round(15 + mu_un + rnorm(n, 0, sqrt(2)), 1)

# normal exposure distribution, but marginally not normal
x2 <- round(15 + mu_g + rnorm(n, 0, sqrt(2)), 1)

# poisson exposure distribution
x3 <- round(pmax(rpois(n, mu_un) + rnorm(n,0,1), 0), 1)

# log of x3
x4 <- log(x3 + 0.001)

# now replicate Naimi's continuous exposures
n_x1 <- 15 + mu_un + rnorm(n, 0, sqrt(2))
# n_x1 <- rnorm(n, 15 + mu_un, 1.5) #
# <- I think this is how they technically did it, but they are the same.

n_x2 <- pmax(rpois(n, mu_un) + rnorm(n,0,1), 0)

# outcome normal exposure distribution, uncorrelated with maternal age
y1 <- rbinom(n, 1, (1  + exp(-(-11.5 + log(1.25) * x1 + log(1.7) * sqrt(mage) + log(1.5) * sqrt(page)
                      log(0.75) * parity2 + log(0.8) * parity3 + log(0.85) * parity4 + log(0.9) *parit

# normal exposure distribution, but marginally not normal, updated intercept from -11.5 to -11.4
y2 <- rbinom(n, 1, (1  + exp(-(-11.4 + log(1.25) * x2 + log(1.7) * sqrt(mage_g) +
                                 log(1.5) * sqrt(page) + log(0.75) * parity2 +
                                 log(0.8) * parity3 + log(0.85) * parity4 +
                                 log(0.9) *parity5)))^(-1))

# outcome poisson exposure distribution, uncorrelated with maternal age
y3 <- rbinom(n, 1, (1 + exp(-(-8.05 + log(1.25) * x3 + log(1.7) * sqrt(mage) +
                                log(1.5) * sqrt(page) + log(0.75) * parity2 +
                                log(0.8) * parity3 + log(0.85) * parity4 +
                                log(0.9) * parity5)))^(-1))

# replicate Naimi's outcomes given continuous exposures
n_y1 <- rbinom(n, 1, (1  + exp(-(-11.5 + log(1.25) * n_x1 + log(1.7) * sqrt(mage) +
                                  log(1.5) * sqrt(page) + log(0.75) * parity2 +
```

```
                                    log(0.8) * parity3 + log(0.85) * parity4 +
                                    log(0.9) * parity5)))^(-1))

  n_y2 <- rbinom(n, 1, (1 + exp(-(-8.05 + log(1.25) * n_x2 + log(1.7) * sqrt(mage) +
                                   log(1.5) * sqrt(page) + log(0.75) * parity2 +
                                   log(0.8) * parity3 + log(0.85) * parity4 +
                                   log(0.9) * parity5)))^(-1))

  # create df with all covariates as output
  data.frame(mage, mage_g, page, parity2, parity3, parity4, parity5,
             x1, x2, x3, x4, n_x1, n_x2, y1, y2, y3, n_y1, n_y2)
}

# to check outcome prevalence with different intercepts
#table1::table1(~ factor(y1) + factor(y2) + factor(y3) + factor(y4) +
#factor(y5) + factor(y6), data = sim_data(n = 10000))
```

Simulations

Do 3000 simulations and generate weights for each simulation, combine each simulation into one long dataframe with weights and list and simulation number. Generate weights with OLS, CBGPS, QB10, QB15, QB20, and OLR.

For X_2, we updated the intercept values (from -11.5 [X_1] to -11.4 [X_2]) to maintain a marginal probability of the outcome of approximately 0.08 with the updated exposure distributions.

We calculated the sIPW denominators using the following regression formula per Naimi et al.'s specifications, where C are the selected confounders, with *mage_g* instead of *mage* when i ϵ {2} and binned exposures instead of X_i when calculating QB weights:

$$E(X_i \mid C) = \beta_1(\text{mage}) + \beta_2(\text{page}) + \beta_3(\text{mage*page}) + \beta_4(\text{parity2}) + \beta_5(\text{parity3}) + \beta_6(\text{parity4}) + \beta_7(\text{parity5})$$

The range, median, and mean of the exposure distributions are in Supplemental Table 1.2.

```
# register clusters (use 7 cores)
registerDoParallel(detectCores() - 1)

# number of reps (will go up to 3000, can tinker to test things)
n = 3000

# now generate weights
# (will generate weights in each simulated dataset individually to speed cbgps process)
sims <- foreach(i = 1:n, .inorder = FALSE, .errorhandling = "remove") %dopar% {
  # first need to generate data and quantile binned exposures
  df <- sim_data(n = 1500) %>%
  mutate(x1_qb10 = as.numeric(cut2(x1, g = 10)),
         x2_qb10 = as.numeric(cut2(x2, g = 10)),
         x3_qb10 = as.numeric(cut2(x3, g = 10)),
         x4_qb10 = as.numeric(cut2(x4, g = 10)),
         x1_qb15 = as.numeric(cut2(x1, g = 15)),
         x2_qb15 = as.numeric(cut2(x2, g = 15)),
         x3_qb15 = as.numeric(cut2(x3, g = 15)),
         x4_qb15 = as.numeric(cut2(x4, g = 15)),
         x1_qb20 = as.numeric(cut2(x1, g = 20)),
```

```r
       x2_qb20 = as.numeric(cut2(x2, g = 20)),
       x3_qb20 = as.numeric(cut2(x3, g = 20)),
       x4_qb20 = as.numeric(cut2(x4, g = 20)))

# start by creating formulas
x1_formula <- formula(x1 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x2_formula <- formula(x2 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
x3_formula <- formula(x3 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x4_formula <- formula(x4 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)

# use WeightIt package to generate OLS and CBGPS weights

# OLS
x1_ols_wts <- weightit(x1_formula, df %>% filter(!is.na(x1)), method = "ps")$weights
x2_ols_wts <- weightit(x2_formula, df %>% filter(!is.na(x2)), method = "ps")$weights
x3_ols_wts <- weightit(x3_formula, df %>% filter(!is.na(x3)), method = "ps")$weights
x4_ols_wts <- weightit(x4_formula, df %>% filter(!is.na(x4)), method = "ps")$weights

#CBGPS
x1_cbgps_wts <- weightit(x1_formula, df %>% filter(!is.na(x1)), method = "cbps",
                         over = FALSE)$weights
x2_cbgps_wts <- weightit(x2_formula, df %>% filter(!is.na(x2)), method = "cbps",
                         over = FALSE)$weights
x3_cbgps_wts <- weightit(x3_formula, df %>% filter(!is.na(x3)), method = "cbps",
                         over = FALSE)$weights
x4_cbgps_wts <- weightit(x4_formula, df %>% filter(!is.na(x4)), method = "cbps",
                         over = FALSE)$weights

# use orm.wt file to create quantile binning and OLR weights

# QB10
x1_qb10_wts <- orm.wt(object = df %>% filter(!is.na(x1)),
                  exposure = "x1_qb10",
                  cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                  parity5") %>%
  unlist()
x2_qb10_wts <- orm.wt(object = df %>% filter(!is.na(x2)),
                  exposure = "x2_qb10",
                  cov_form = "~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
                  parity5") %>%
  unlist()
x3_qb10_wts <- orm.wt(object = df %>% filter(!is.na(x3)),
                  exposure = "x3_qb10",
                  cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                  parity5") %>%
  unlist()
x4_qb10_wts <- orm.wt(object = df %>% filter(!is.na(x4)),
                  exposure = "x4_qb10",
                  cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                  parity5") %>%
  unlist()

# QB15
```

```
x1_qb15_wts <- orm.wt(object = df %>% filter(!is.na(x1)),
                     exposure = "x1_qb15",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x2_qb15_wts <- orm.wt(object = df %>% filter(!is.na(x2)),
                     exposure = "x2_qb15",
                     cov_form = "~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x3_qb15_wts <- orm.wt(object = df %>% filter(!is.na(x3)),
                     exposure = "x3_qb15",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x4_qb15_wts <- orm.wt(object = df %>% filter(!is.na(x4)),
                     exposure = "x4_qb15",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()

# QB20
x1_qb20_wts <- orm.wt(object = df %>% filter(!is.na(x1)),
                     exposure = "x1_qb20",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x2_qb20_wts <- orm.wt(object = df %>% filter(!is.na(x2)),
                     exposure = "x2_qb20",
                     cov_form = "~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x3_qb20_wts <- orm.wt(object = df %>% filter(!is.na(x3)),
                     exposure = "x3_qb20",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x4_qb20_wts <- orm.wt(object = df %>% filter(!is.na(x4)),
                     exposure = "x4_qb20",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()

# OLR
x1_olr_wts <- orm.wt(object = df %>% filter(!is.na(x1)),
                     exposure = "x1",
                     cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                     parity5") %>%
  unlist()
x2_olr_wts <- orm.wt(object = df %>% filter(!is.na(x2)),
                     exposure = "x2",
                     cov_form = "~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
                     parity5") %>%
```

```r
    unlist()
  x3_olr_wts <- orm.wt(object = df %>% filter(!is.na(x3)),
                      exposure = "x3",
                      cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                      parity5") %>%
    unlist()
  x4_olr_wts <- orm.wt(object = df %>% filter(!is.na(x4)),
                      exposure = "x4",
                      cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
                      parity5") %>%
    unlist()

  # create final dataframe
  data <- data.frame(i, df,
                     x1_ols_wts, x2_ols_wts, x3_ols_wts, x4_ols_wts,
                     x1_cbgps_wts, x2_cbgps_wts, x3_cbgps_wts, x4_cbgps_wts,
                     x1_qb10_wts, x2_qb10_wts, x3_qb10_wts, x4_qb10_wts,
                     x1_qb15_wts, x2_qb15_wts, x3_qb15_wts, x4_qb15_wts,
                     x1_qb20_wts, x2_qb20_wts, x3_qb20_wts, x4_qb20_wts,
                     x1_olr_wts, x2_olr_wts, x3_olr_wts, x4_olr_wts)
}

# save simulation output
Save(sims)
```

```r
# load simulation data
Load(sims)

# make it a dataframe
df <- sims %>% bind_rows(.id = "i")

# create a new dataset with 4.5 million rows to simluate the truth
set.seed(1111)
df_msm_sim <- sim_data(n = nrow(df))
```

```r
# Marginal Structural Model "Truth"
getMeanProb_MSM <- function(dat, val, xnum) {
    # returns the mean outcome probability (that is, an estimate of E[Y(t)]) at exposure value [val]
    if (xnum == "x1") {
        alpha <- -11.5
        magevar <- "mage"
    } else if (xnum == "x2") {
        alpha <- -11.4
        magevar <- "mage_g"
    } else if (xnum == "x3") {
        alpha <- -8.05
        magevar <- "mage"
    }

    lp <-
        alpha +
        log(1.25) * (val) +
        log(1.7) * sqrt(dat[[magevar]])+
        log(1.5) * sqrt(dat$page) +
```

```r
        log(0.75) * dat$parity2 +
        log(0.8) * dat$parity3 +
        log(0.85) * dat$parity4 +
        log(0.9) * dat$parity5
    prob <- plogis(lp)
    mean(prob)
}

# run it with parallel processing

# register clusters (use 3 cores because 3 processes)
registerDoParallel(3)

# now generate weights
msm_truths <- foreach(i = 1:3, .inorder = FALSE, .errorhandling = "remove") %dopar% {
  #exposures
  exps <- c("x1", "x2", "x3")

  # each is a vector of estimates of E[Y(t)]'s: one for each unique t in the original dataset
  # (excluding repeats)
  probs <- lapply(unique(df_msm_sim[[exps[i]]]),
                  function(x) getMeanProb_MSM(df_msm_sim, x, xnum = exps[i]))
}

# save it
Save(msm_truths)
```

```r
# pull in MSM truth, because it  will be too big to run each time
Load(msm_truths)
x1_truth_key <- tibble(x1 = unique(df_msm_sim$x1), prob_x1 = unlist(msm_truths[1]))
x2_truth_key <- tibble(x2 = unique(df_msm_sim$x2), prob_x2 = unlist(msm_truths[2]))
x3_truth_key <- tibble(x3 = unique(df_msm_sim$x3), prob_x3 = unlist(msm_truths[3]))

# now need to link truths with exposure they match
x_truths <- df_msm_sim %>%
  select(x1, x2, x3)

# put in truths
x_truths <- left_join(x_truths, x1_truth_key, by = "x1")
x_truths <- left_join(x_truths, x2_truth_key, by = "x2")
x_truths <- left_join(x_truths, x3_truth_key, by = "x3")

# show that there are no non-missing values of prob_x1, prob_x2, and prob_x3 after joining
sum(is.na(x_truths$prob_x1), is.na(x_truths$prob_x2), is.na(x_truths$prob_x3))
```

```
[1] 0
```

```r
sum(!is.na(x_truths$prob_x1), !is.na(x_truths$prob_x2), !is.na(x_truths$prob_x3))/3
```

```
[1] 4500000
```

```r
# Our MSM: logit{E[Y(t)]}= b0 + b1*t
#     The warning is OK!
msm_x1 <- glm(x_truths$prob_x1 ~ x_truths$x1, family= binomial)
```

```
true_x1 <- coef(msm_x1)["x_truths$x1"] %>% unname()

msm_x2 <- glm(x_truths$prob_x2 ~ x_truths$x2, family= binomial)
```

```
true_x2 <- coef(msm_x2)["x_truths$x2"] %>% unname()

msm_x3 <- glm(x_truths$prob_x3 ~ x_truths$x3, family= binomial)
```

```
true_x3 <- coef(msm_x3)["x_truths$x3"] %>% unname()

# Austin, 2018 approach to finding the truth
# find the true probability, across all 4.5 million observations at each decile (seq(0.1, 0.9, 0.1))
# the the expected probabilities are found below in the bias chunk
getMeanProb <- function(dat, val, xnum) {
    if (xnum == 1) {
        alpha <- -11.5
        magevar <- "mage"
    } else if (xnum == 2) {
        alpha <- -11.4
        magevar <- "mage_g"
    } else if (xnum == 3) {
        alpha <- -8.05
        magevar <- "mage"
    }
    lp <- alpha +
        log(1.25) * (val) +
        log(1.7) * sqrt(dat[[magevar]])+
        log(1.5) * sqrt(dat$page) +
        log(0.75) * dat$parity2 +
        log(0.8) * dat$parity3 +
        log(0.85) * dat$parity4 +
        log(0.9) * dat$parity5
    # p = (1 + 1/odds)^(-1)
    prob <- (1 + 1/exp(lp))^(-1)
    mean(prob)
}

# x1
true2_x1_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x1, .x), xnum = 1))

# x2
true2_x2_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x2, .x), xnum = 2))

# x3
true2_x3_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x3, .x), xnum = 3))
```

Supplemental Table 1.1 - Dose Response Declies

Decile	X1	X2	X3
1	15.1	21.4	0.0
2	15.7	22.3	0.5
3	16.2	22.9	0.9
4	16.6	23.4	1.4
5	17.0	23.9	1.8
6	17.3	24.3	2.3
7	17.7	24.7	2.8
8	18.2	25.3	3.4
9	18.8	25.9	4.3

```r
# create deciles quantiles for each Austin approach
x1_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x1, .x))

x2_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x2, .x))

x3_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x3, .x))

# print table of decile values
dec_tab <- tibble(Decile = c(1:9),
                  X1 = x1_quants,
                  X2 = x2_quants,
                  X3 = x3_quants)
kable(dec_tab) %>%
  kable_classic(html_font = "Arial", full_width = FALSE)
```

Recreating Distributions from Naimi et al.

Supplemental Table 1.2 - Simulation Descriptive Statistics

```r
# recreate table 1
tab1 <- tibble(`Variable (Distribution)` = c("Maternal Age (normal)",
                                             "Maternal Age (skewed)",
                                             "Paternal Age (normal)",
                                             "Parity (Poisson)",
                                             "2",
                                             "3",
                                             "4",
                                             "5+",
                                             "X1 (normal, naimi)",
                                             "X2 (normal, skewed)",
                                             "X3 (Poisson, naimi)",
                                             #"X4 (Poisson, log(naimi))",
                                             "Y1 (Bernoulli, naimi)",
                                             "Y2 (Bernoulli, skewed)",
                                             "Y3 (Bernoulli, naimi)",
                                             "Naimi Homoscedastic X",
                                             "Naimi Heteroscedastic X",
                                             "Naimi Homoscedastic Y",
                                             "Naimi Heteroscedastic Y"),
```

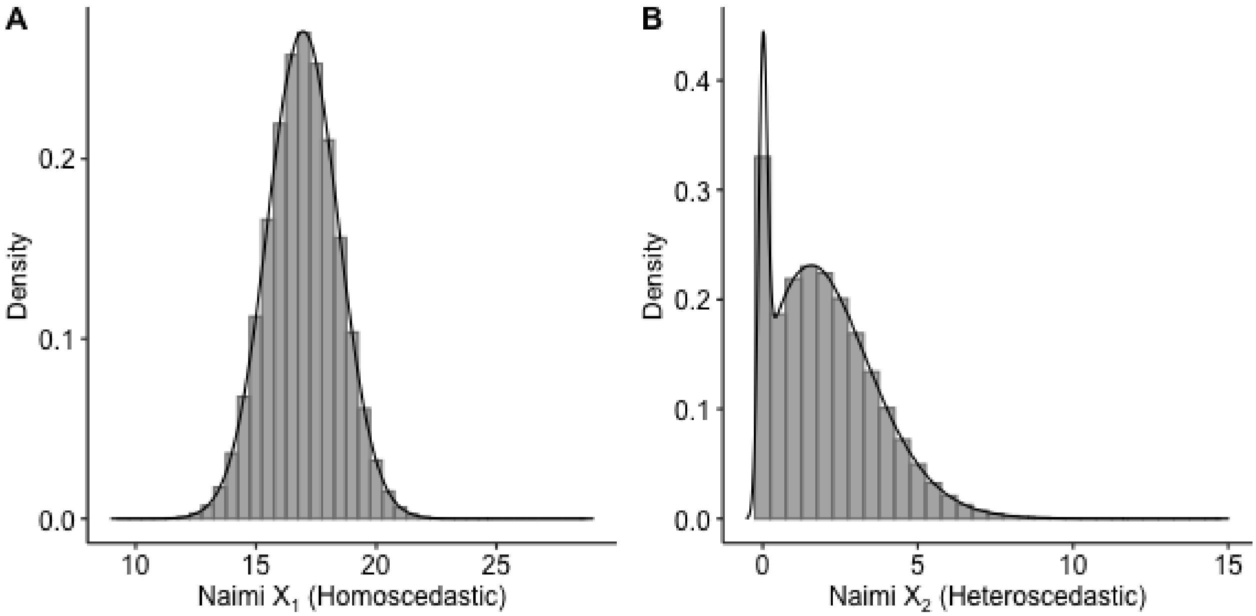
```r
               Mean = c(mean(df$mage),
                        mean(df$mage_g),
                        mean(df$page),
                        NA,
                        mean(df$parity2),
                        mean(df$parity3),
                        mean(df$parity4),
                        mean(df$parity5),
                        mean(df$x1),
                        mean(df$x2),
                        mean(df$x3, na.rm = TRUE),
                        #mean(df$x4, na.rm = TRUE),
                        mean(df$y1),
                        mean(df$y2),
                        mean(df$y3, na.rm = TRUE),
                        mean(df$n_x1),
                        mean(df$n_x2, na.rm = TRUE),
                        mean(df$n_y1),
                        mean(df$n_y2, na.rm = TRUE)),
               Variance = c(var(df$mage),
                        var(df$mage_g),
                        var(df$page),
                        NA,
                        var(df$parity2),
                        var(df$parity3),
                        var(df$parity4),
                        var(df$parity5),
                        var(df$x1),
                        var(df$x2),
                        var(df$x3, na.rm = TRUE),
                        #var(df$x4, na.rm = TRUE),
                        var(df$y1),
                        var(df$y2),
                        var(df$y3, na.rm = TRUE),
                        var(df$n_x1),
                        var(df$n_x2, na.rm = TRUE),
                        var(df$n_y1),
                        var(df$n_y2, na.rm = TRUE)))

# table 1
kable(tab1, digits = 2) %>%
  kable_classic(html_font = "Arial", full_width = FALSE) %>%
  add_indent(c(5:8))
```

Supplemental Figure 1.1 - Continuous Exposure

```r
# now create plot
naimix1 <- ggplot(df, aes(x = n_x1)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste("Naimi ", X[1], " (Homoscedastic)")),
```

Variable (Distribution)	Mean	Variance
Maternal Age (normal)	29.84	21.60
Maternal Age (skewed)	30.07	15.46
Paternal Age (normal)	32.52	30.47
Parity (Poisson)		
2	0.24	0.18
3	0.07	0.07
4	0.02	0.02
5+	0.02	0.02
X1 (normal, naimi)	16.96	2.16
X2 (normal, skewed)	23.74	3.37
X3 (Poisson, naimi)	2.04	2.67
Y1 (Bernoulli, naimi)	0.08	0.07
Y2 (Bernoulli, skewed)	0.29	0.20
Y3 (Bernoulli, naimi)	0.09	0.08
Naimi Homoscedastic X	16.96	2.16
Naimi Heteroscedastic X	2.04	2.67
Naimi Homoscedastic Y	0.08	0.07
Naimi Heteroscedastic Y	0.09	0.08

```
                     limits = c(9, 29), breaks = c(10, 15, 20, 25)) +
  theme

naimix2 <- ggplot(df, aes(x = n_x2)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste("Naimi ", X[2], " (Heteroscedastic)")),
                     limits = c(-0.5, 15), breaks = c(0, 5, 10, 15)) +
  theme

# recreate figure 1
ggarrange(naimix1, naimix2,
          labels = c("A", "B"))
```

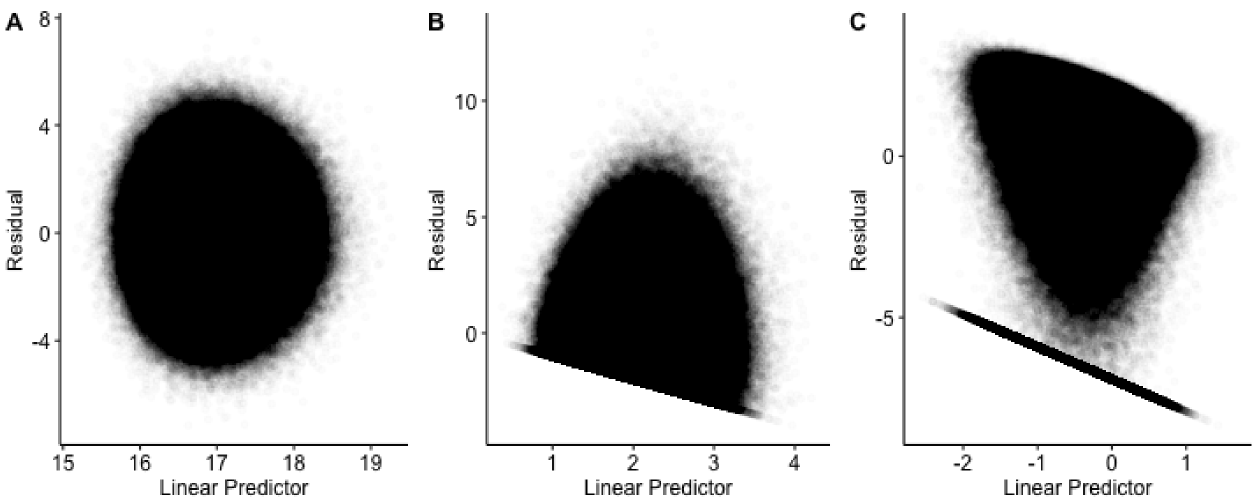


Panel A) Homoscedastic Continuous Exposure, Panel B) Heteroscedastic Continuous Exposure

Supplemental Figure 1.2 - Continuous Exposure

```
# will run ols regression on df
ols_x1_n <- ols(n_x1 ~ + mage + page + mage*page + parity2 + parity3 + parity4 +
                  parity5, data = df)
ols_x2_n <- ols(n_x2 ~ + mage + page + mage*page + parity2 + parity3 + parity4 +
                  parity5, data = df) # added 0.001 to avoid -Inf when logging
ols_x2_n_log <- ols(log(n_x2 + 0.001) ~ + mage + page + mage*page + parity2 + parity3 + parity4 +
                      parity5, data = df) # added 0.001 to avoid -Inf when logging

# linear predictors
preds_x1_n <- predict(ols_x1_n)
preds_x2_n <- predict(ols_x2_n)
preds_x2_n_log <- predict(ols_x2_n_log)

# residuals
res_x1_n <- residuals(ols_x1_n)
res_x2_n <- residuals(ols_x2_n)
res_x2_n_log <- residuals(ols_x2_n_log)

# now plot
x1_plot_n <- ggplot() +
  geom_point(aes(x = preds_x1_n, y = res_x1_n), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme

x2_plot_n <- ggplot() +
  geom_point(aes(x = preds_x2_n, y = res_x2_n), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
```

```
  theme

x2_plot_n_log <- ggplot() +
  geom_point(aes(x = preds_x2_n_log, y = res_x2_n_log), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme

# combine
ggarrange(x1_plot_n, x2_plot_n, x2_plot_n_log,
          nrow = 1,
          labels = c("A", "B", "C"))
```



Panel A) Homoscedastic Continuous Exposure, Panel B) Heteroscedastic Continuous Exposure, Panel C) Heteroscadastic log of Continuous Exposure

Updated Distributions

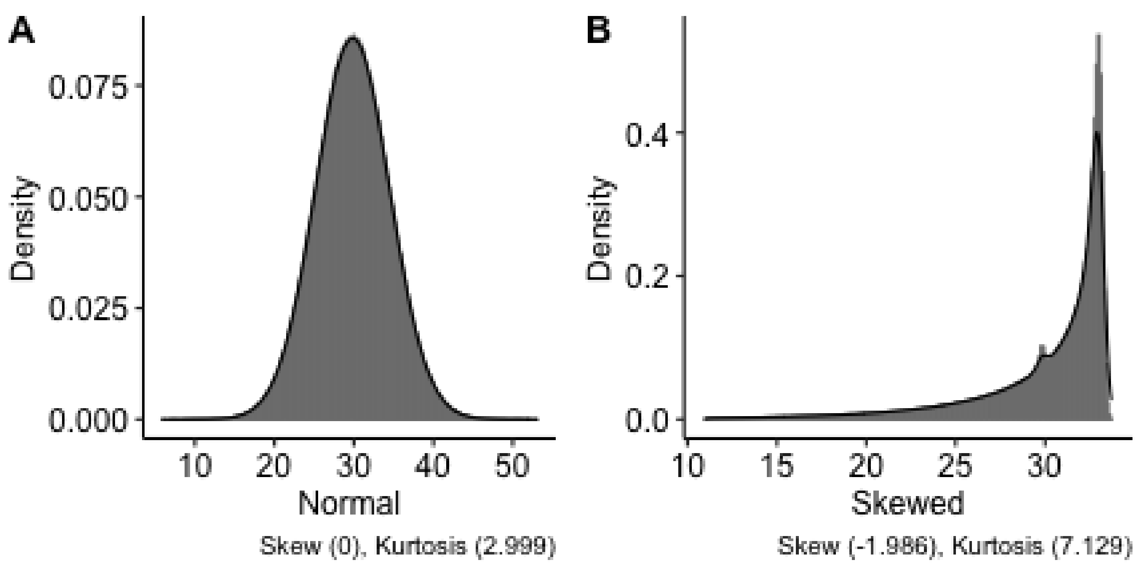
Supplemental Table 1.3 - Updated Exposure Levels

```
# get number of exposure levels across simulations
exp_levels <- function(data) {
  x1 <- n_distinct(data$x1)
  x2 <- n_distinct(data$x2)
  x3 <- n_distinct(data$x3)
  #x4 <- n_distinct(data$x4)

  data.frame(x1, x2, x3)
}

suptab2 <- map_df(sims, ~ exp_levels(.x)) %>% summary()
kable(suptab2) %>%
  kable_classic(html_font = "Arial", full_width = FALSE)
```

	x1	x2	x3
	Min. :77.00	Min. : 96.0	Min. :67.00
	1st Qu.:83.00	1st Qu.:105.0	1st Qu.:75.00
	Median :85.00	Median :107.0	Median :76.00
	Mean :85.08	Mean :106.7	Mean :76.36
	3rd Qu.:87.00	3rd Qu.:109.0	3rd Qu.:78.00
	Max. :94.00	Max. :121.0	Max. :85.00

Supplemental Figure 1.3 - Maternal Age Distributions

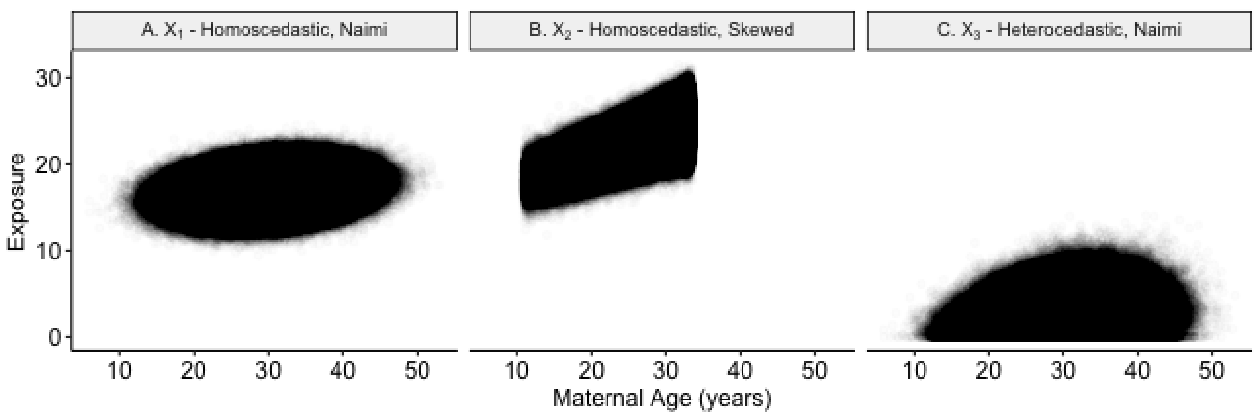
```
mage_hist <- df %>%
  ggplot(aes(x = mage)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  xlab("Normal") +
  labs(caption = paste0("Skew (", round(moments::skewness(df$mage), 3), "), Kurtosis (",
                        round(moments::kurtosis(df$mage), 3), ")")) +
  theme

mage_g_hist <- df %>%
  ggplot(aes(x = mage_g)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  xlab("Skewed") +
  labs(caption = paste0("Skew (", round(moments::skewness(df$mage_g), 3), "), Kurtosis (",
                        round(moments::kurtosis(df$mage_g), 3), ")")) +
  theme

ggarrange(mage_hist, mage_g_hist,
          labels = c("A", "B"))
```



Supplemental Figure 3 shows the updated marginal distributions of maternal age (`mage`) and skewed maternal age (`mage_g`). Despite similar means and variances in `mage` and `mage_g`, `mage` is normally distributed whereas `mage_g` is significantly left-skewed.

Supplemental Figure 1.4 - Exposure-Covariate Correlations

```
df %>%
  select(mage, mage_g, x1:x3) %>%
  pivot_longer(x1:x3) %>%
  mutate(mage_fin = ifelse(name %in% c("x1", "x3"), mage, mage_g)) %>%
  mutate(name = factor(name,
                       labels = c(expression(paste("A. ", X[1], " - Homoscedastic, Naimi")),
                                  expression(paste("B. ", X[2], " - Homoscedastic, Skewed")),
                                  expression(paste("C. ", X[3], " - Heterocedastic, Naimi"))))) %>%
                                  #expression(paste("D. ", X[4], " - Heterocedastic, log(Naimi + 0.001)
  ggplot(aes(x = mage_fin, y = value)) +
  facet_wrap(~ name, labeller = "label_parsed") +
  geom_point(alpha = 0.01) +
  ylab("Exposure") +
  xlab("Maternal Age (years)") +
  theme
```



Supplemental Figure 4 shows the relationships between maternal age (`mage`) or skewed maternal age (`mage_g`) and the generated exposure scenarios, which confirms increased correlations in the correlated case (Panel B [X_2]).

Marginal and Conditional Exposure Distributions

Figure 1 - Marginal and Conditional Exposure Distribution

```
# Marginal Exposure Distribution

# now create plot
homoscedasticx1 <- ggplot(df, aes(x = x1)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[1]))) +
```

```r
  # labs(caption = paste0("Skew (", round(moments::skewness(df$x1), 3),
  #       "), Kurtosis (", round(moments::kurtosis(df$x1), 3), ")")) +
  theme +
  theme(text = element_text(size = 8.5))

heteroscedasticx2 <- ggplot(df, aes(x = x2)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[2]))) +
  # labs(caption = paste0("Skew (", round(moments::skewness(df$x2), 3),
  #       "), Kurtosis (", round(moments::kurtosis(df$x2), 3), ")")) +
  theme +
  theme(text = element_text(size = 8.5))

homoscedasticx3 <- ggplot(df, aes(x = x3)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
  geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[3]))) +
  # labs(caption = "  ") +
  theme +
  theme(text = element_text(size = 8.5))

# heteroscedasticx4 <- ggplot(df, aes(x = x4)) +
#   geom_histogram(aes(y = ..density..), binwidth = 0.1, alpha = 0.5, color = "grey50") +
#   geom_density(adjust = 10) +
#   ylab("Density") +
#   scale_x_continuous(name = expression(paste(X[4], " (C: Non-Normal; M: Non-Normal - log)"))) +
#   theme

# Conditional Exposure Distribution

# will run ols regression on df
ols_x1 <- ols(x1 ~ + mage + page + mage*page + parity2 + parity3 + parity4 +
                parity5, data = df)
ols_x2 <- ols(x2 ~ + mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
                parity5, data = df)
ols_x3 <- ols(x3 ~ + mage + page + mage*page + parity2 + parity3 + parity4 +
                parity5, data = df)
# ols_x4 <- ols(x4 ~ + mage + page + mage*page + parity2 + parity3 + parity4 + parity5, data = df)

# linear predictors
preds_x1 <- predict(ols_x1)
preds_x2 <- predict(ols_x2)
preds_x3 <- predict(ols_x3)
# preds_x4 <- predict(ols_x4)

# residuals
res_x1 <- residuals(ols_x1)
res_x2 <- residuals(ols_x2)
res_x3 <- residuals(ols_x3)
# res_x4 <- residuals(ols_x4)
```

```r
# now plot
x1_plot <- ggplot() +
  geom_point(aes(x = preds_x1, y = res_x1), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme +
  theme(text = element_text(size = 8.5))

x2_plot <- ggplot() +
  geom_point(aes(x = preds_x2, y = res_x2), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme +
  theme(text = element_text(size = 8.5))

x3_plot <- ggplot() +
  geom_point(aes(x = preds_x3, y = res_x3), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme +
  theme(text = element_text(size = 8.5))

# x4_plot <- ggplot() +
#   geom_point(aes(x = preds_x4, y = res_x4), alpha = 0.01) +
#   ylab("Residual") +
#   xlab("Linear Predictor") +
#   theme +
#   theme(text = element_text(size = 8))

# combine
ggarrange(homoscedasticx1, heteroscedasticx2,
          homoscedasticx3,
          x1_plot, x2_plot, x3_plot,
          labels = c("A)", "B)", "C)", "D)", "E)", "F)"),
          nrow = 2, ncol = 3)
# save plot
ggsave("figs/fig1.tiff", width = 7, height = 5)
```

Stabilized Inverse Probability Weight Assessments

```r
# make dataframe that gives the mean IPW weights for each simulation
# (will then take mean, min, and max of those)
mean_wts <- df %>%
  select(i, x1_ols_wts:x4_olr_wts) %>%
  group_by(i) %>%
  summarise(x1_ols_wts = mean(x1_ols_wts),
            x1_cbgps_wts = mean(x1_cbgps_wts),
            x1_qb10_wts = mean(x1_qb10_wts),
            x1_qb15_wts = mean(x1_qb15_wts),
            x1_qb20_wts = mean(x1_qb20_wts),
            x1_olr_wts = mean(x1_olr_wts),
```

```
            x2_ols_wts = mean(x2_ols_wts),
            x2_cbgps_wts = mean(x2_cbgps_wts),
            x2_qb10_wts = mean(x2_qb10_wts),
            x2_qb15_wts = mean(x2_qb15_wts),
            x2_qb20_wts = mean(x2_qb20_wts),
            x2_olr_wts = mean(x2_olr_wts),
            x3_ols_wts = mean(x3_ols_wts),
            x3_cbgps_wts = mean(x3_cbgps_wts),
            x3_qb10_wts = mean(x3_qb10_wts),
            x3_qb15_wts = mean(x3_qb15_wts),
            x3_qb20_wts = mean(x3_qb20_wts),
            x3_olr_wts = mean(x3_olr_wts))

# have to get mean (min, max) of weights from different exposure scenarios
tab2 <- tibble(Method = c("Ordinary least squares",
                          "Covariate balancing generalized propensity score",
                          "Quantile binning categories",
                          "10",
                          "15",
                          "20",
                          "Ordinal logistic regression"),
               `Mean (min, max)` = c(paste0(round(mean(mean_wts$x1_ols_wts), 2), " (",
                                           round(min(mean_wts$x1_ols_wts), 2), ", ",
                                           round(max(mean_wts$x1_ols_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x1_cbgps_wts), 2), " (",
                                           round(min(mean_wts$x1_cbgps_wts), 2), ", ",
                                           round(max(mean_wts$x1_cbgps_wts), 2), ")"),
                                    NA,
                                    paste0(round(mean(mean_wts$x1_qb10_wts), 2), " (",
                                           round(min(mean_wts$x1_qb10_wts), 2), ", ",
                                           round(max(mean_wts$x1_qb10_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x1_qb15_wts), 2), " (",
                                           round(min(mean_wts$x1_qb15_wts), 2), ", ",
                                           round(max(mean_wts$x1_qb15_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x1_qb20_wts), 2), " (",
                                           round(min(mean_wts$x1_qb20_wts), 2), ", ",
                                           round(max(mean_wts$x1_qb20_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x1_olr_wts), 2), " (",
                                           round(min(mean_wts$x1_olr_wts), 2), ", ",
                                           round(max(mean_wts$x1_olr_wts), 2), ")")),
               `Mean (min, max) ` = c(paste0(round(mean(mean_wts$x2_ols_wts), 2), " (",
                                           round(min(mean_wts$x2_ols_wts), 2), ", ",
                                           round(max(mean_wts$x2_ols_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x2_cbgps_wts), 2), " (",
                                           round(min(mean_wts$x2_cbgps_wts), 2), ", ",
                                           round(max(mean_wts$x2_cbgps_wts), 2), ")"),
                                    NA,
                                    paste0(round(mean(mean_wts$x2_qb10_wts), 2), " (",
                                           round(min(mean_wts$x2_qb10_wts), 2), ", ",
                                           round(max(mean_wts$x2_qb10_wts), 2), ")"),
                                    paste0(round(mean(mean_wts$x2_qb15_wts), 2), " (",
                                           round(min(mean_wts$x2_qb15_wts), 2), ", ",
                                           round(max(mean_wts$x2_qb15_wts), 2), ")"),
```

```r
                                paste0(round(mean(mean_wts$x2_qb20_wts), 2), " (",
                                       round(min(mean_wts$x2_qb20_wts), 2), ", ",
                                       round(max(mean_wts$x2_qb20_wts), 2), ")"),
                                paste0(round(mean(mean_wts$x2_olr_wts), 2), " (",
                                       round(min(mean_wts$x2_olr_wts), 2), ", ",
                                       round(max(mean_wts$x2_olr_wts), 2), ")")),
                `Mean (min, max)  ` = c(paste0(round(mean(mean_wts$x3_ols_wts), 2), " (",
                                       round(min(mean_wts$x3_ols_wts), 2), ", ",
                                       round(max(mean_wts$x3_ols_wts), 2), ")"),
                                paste0(round(mean(mean_wts$x3_cbgps_wts), 2), " (",
                                       round(min(mean_wts$x3_cbgps_wts), 2), ", ",
                                       round(max(mean_wts$x3_cbgps_wts), 2), ")"),
                                NA,
                                paste0(round(mean(mean_wts$x3_qb10_wts), 2), " (",
                                       round(min(mean_wts$x3_qb10_wts), 2), ", ",
                                       round(max(mean_wts$x3_qb10_wts), 2), ")"),
                                paste0(round(mean(mean_wts$x3_qb15_wts), 2), " (",
                                       round(min(mean_wts$x3_qb15_wts), 2), ", ",
                                       round(max(mean_wts$x3_qb15_wts), 2), ")"),
                                paste0(round(mean(mean_wts$x3_qb20_wts), 2), " (",
                                       round(min(mean_wts$x3_qb20_wts), 2), ", ",
                                       round(max(mean_wts$x3_qb20_wts), 2), ")"),
                                paste0(round(mean(mean_wts$x3_olr_wts), 2), " (",
                                       round(min(mean_wts$x3_olr_wts), 2), ", ",
                                       round(max(mean_wts$x3_olr_wts), 2), ")"))#,
                # `Mean (min, max)   ` = c(paste0(round(mean(df$x4_ols_wts), 2), " (",
                #                               round(min(df$x4_ols_wts), 2), ", ",
                #                               round(max(df$x4_ols_wts), 2), ")"),
                #                        paste0(round(mean(df$x4_cbgps_wts), 2), " (",
                #                               round(min(df$x4_cbgps_wts), 2), ", ",
                #                               round(max(df$x4_cbgps_wts), 2), ")"),
                #                        NA,
                #                        paste0(round(mean(df$x4_qb10_wts), 2), " (",
                #                               round(min(df$x4_qb10_wts), 2), ", ",
                #                               round(max(df$x4_qb10_wts), 2), ")"),
                #                        paste0(round(mean(df$x4_qb15_wts), 2), " (",
                #                               round(min(df$x4_qb15_wts), 2), ", ",
                #                               round(max(df$x4_qb15_wts), 2), ")"),
                #                        paste0(round(mean(df$x4_qb20_wts), 2), " (",
                #                               round(min(df$x4_qb20_wts), 2), ", ",
                #                               round(max(df$x4_qb20_wts), 2), ")"),
                #                        paste0(round(mean(df$x4_olr_wts), 2), " (",
                #                               round(min(df$x4_olr_wts), 2), ", ",
                #                               round(max(df$x4_olr_wts), 2), ")"))
                )

# # table 2
# kable(tab2) %>%
#   kable_classic(html_font = "Arial", full_width = FALSE) %>%
#   add_header_above(c("", "X1" = 1, "X2" = 1, "X3" = 1), bold = TRUE) %>%
#   add_header_above(c("Marginally", "Normal" = 1, "Non-Normal" = 1,
#                      "Non-Normal" = 1), bold = TRUE) %>%
#   add_header_above(c("Conditionally", "Normal" = 1, "Normal" = 1,
```

```
#                       "Non-Normal" = 1), bold = TRUE) %>%
#   # add_header_above(c("Expsoure Transformation", "None" = 3,
#   #                     "log(X + 0.001)" = 1), bold = TRUE) %>%
#   add_indent(c(4:6))
```

Covariate Balance

```
# will need to calculate number of covariates with correlation greated than 0.1 in all exposure scenari
# start with a function (ignore QB for now)

covbal_func <- function(data){
  # simulation number
  i <- data$i[1]

  # start with formulas for different exposures
  x1_formula <- formula(x1 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
  x2_formula <- formula(x2 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
  x3_formula <- formula(x3 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
  x4_formula <- formula(x4 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)

  # now calculate balance
  bal_tab_x1 <- bal.tab(x1_formula, data = data,
        weights = list(OLS = "x1_ols_wts",
                       CBGPS = "x1_cbgps_wts",
                       OLR = "x1_olr_wts"),
        stats = c("c"),
        un = TRUE, thresholds = c(cor = .1))
  bal_tab_x2 <- bal.tab(x2_formula, data = data,
        weights = list(OLS = "x2_ols_wts",
                       CBGPS = "x2_cbgps_wts",
                       OLR = "x2_olr_wts"),
        stats = c("c"),
        un = TRUE, thresholds = c(cor = .1))
  bal_tab_x3 <- bal.tab(x3_formula, data = data,
        weights = list(OLS = "x3_ols_wts",
                       CBGPS = "x3_cbgps_wts",
                       OLR = "x3_olr_wts"),
        stats = c("c"),
        un = TRUE, thresholds = c(cor = .1))
  bal_tab_x4 <- bal.tab(x4_formula, data = data,
        weights = list(OLS = "x4_ols_wts",
                       CBGPS = "x4_cbgps_wts",
                       OLR = "x4_olr_wts"),
        stats = c("c"),
        un = TRUE, thresholds = c(cor = .1))

  # now calculate quantile binning correlations

  # qb10
  x1_qb10form <- formula(x1_qb10 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
  x2_qb10form <- formula(x2_qb10 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
  x3_qb10form <- formula(x3_qb10 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
```

```
x4_qb10form <- formula(x4_qb10 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)

bal_tab_x1_qb10 <- bal.tab(x1_qb10form, data = data, weights = "x1_qb10_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x2_qb10 <- bal.tab(x2_qb10form, data = data, weights = "x2_qb10_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x3_qb10 <- bal.tab(x3_qb10form, data = data, weights = "x3_qb10_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x4_qb10 <- bal.tab(x4_qb10form, data = data, weights = "x4_qb10_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))

# qb15
x1_qb15form <- formula(x1_qb15 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x2_qb15form <- formula(x2_qb15 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
x3_qb15form <- formula(x3_qb15 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x4_qb15form <- formula(x4_qb15 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)

bal_tab_x1_qb15 <- bal.tab(x1_qb15form, data = data, weights = "x1_qb15_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x2_qb15 <- bal.tab(x2_qb15form, data = data, weights = "x2_qb15_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x3_qb15 <- bal.tab(x3_qb15form, data = data, weights = "x3_qb15_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x4_qb15<- bal.tab(x4_qb15form, data = data, weights = "x4_qb15_wts", stats = c("c"),
                          un = TRUE, thresholds = c(cor = .1))

# qb20
x1_qb20form <- formula(x1_qb20 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x2_qb20form <- formula(x2_qb20 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
x3_qb20form <- formula(x3_qb20 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
x4_qb20form <- formula(x4_qb20 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)

bal_tab_x1_qb20 <- bal.tab(x1_qb20form, data = data, weights = "x1_qb20_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x2_qb20 <- bal.tab(x2_qb20form, data = data, weights = "x2_qb20_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x3_qb20 <- bal.tab(x3_qb20form, data = data, weights = "x3_qb20_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))
bal_tab_x4_qb20 <- bal.tab(x4_qb20form, data = data, weights = "x4_qb20_wts", stats = c("c"),
                           un = TRUE, thresholds = c(cor = .1))

# now combine into output dataframe
data_frame(i,
           x1_uw = sum(bal_tab_x1$Balance$Corr.Un < 0.1),
           x1_ols = bal_tab_x1$Balanced.correlations[2, 1],
           x1_cbgps = bal_tab_x1$Balanced.correlations[2, 2],
           x1_qb10 = bal_tab_x1_qb10$Balanced.correlations[2, 1],
           x1_qb15 = bal_tab_x1_qb15$Balanced.correlations[2, 1],
           x1_qb20 = bal_tab_x1_qb20$Balanced.correlations[2, 1],
           x1_olr = bal_tab_x1$Balanced.correlations[2, 3],
           x2_uw = sum(bal_tab_x2$Balance$Corr.Un < 0.1),
           x2_ols = bal_tab_x2$Balanced.correlations[2, 1],
           x2_cbgps = bal_tab_x2$Balanced.correlations[2, 2],
```

```
             x2_qb10 = bal_tab_x2_qb10$Balanced.correlations[2, 1],
             x2_qb15 = bal_tab_x2_qb15$Balanced.correlations[2, 1],
             x2_qb20 = bal_tab_x2_qb20$Balanced.correlations[2, 1],
             x2_olr = bal_tab_x2$Balanced.correlations[2, 3],
             x3_uw = sum(bal_tab_x3$Balance$Corr.Un < 0.1),
             x3_ols = bal_tab_x3$Balanced.correlations[2, 1],
             x3_cbgps = bal_tab_x3$Balanced.correlations[2, 2],
             x3_qb10 = bal_tab_x3_qb10$Balanced.correlations[2, 1],
             x3_qb15 = bal_tab_x3_qb15$Balanced.correlations[2, 1],
             x3_qb20 = bal_tab_x3_qb20$Balanced.correlations[2, 1],
             x3_olr = bal_tab_x3$Balanced.correlations[2, 3],
             x4_uw = sum(bal_tab_x4$Balance$Corr.Un < 0.1),
             x4_ols = bal_tab_x4$Balanced.correlations[2, 1],
             x4_cbgps = bal_tab_x4$Balanced.correlations[2, 2],
             x4_qb10 = bal_tab_x4_qb10$Balanced.correlations[2, 1],
             x4_qb15 = bal_tab_x1_qb15$Balanced.correlations[2, 1],
             x4_qb20 = bal_tab_x1_qb20$Balanced.correlations[2, 1],
             x4_olr = bal_tab_x4$Balanced.correlations[2, 3])
}

# # get covariate balance across all simulations
# covbal <- map_df(sims[1:10], ~ covbal_func(.x))

# run in parallel with furrr
plan(multisession, workers = 7)
covbal <- future_map_dfr(sims, ~ covbal_func(.x))

# save
Save(covbal)
```

```
# load covbal
Load(covbal)

# now make table of mean squared error, which is the mean of the squared biases (or errors)
covbal_tab <- tibble(Method = c("Unweighted",
                                "Ordinary least squares",
                          "Covariate balancing generalized propensity score",
                          "Quantile binning categories",
                          "10",
                          "15",
                          "20",
                          "Ordinal logistic regression"),
               `Mean (min, max)` = c(paste0(round(mean(covbal$x1_uw), 2), " (",
                                            min(covbal$x1_uw), ", ",
                                            max(covbal$x1_uw), ")"),
                                     paste0(round(mean(covbal$x1_ols), 2), " (",
                                            min(covbal$x1_ols), ", ",
                                            max(covbal$x1_ols), ")"),
                                     paste0(round(mean(covbal$x1_cbgps), 2), " (",
                                            min(covbal$x1_cbgps), ", ",
                                            max(covbal$x1_cbgps), ")"),
                                     NA,
                                     paste0(round(mean(covbal$x1_qb10), 2), " (",
                                            min(covbal$x1_qb10), ", ",
```

```
                                             max(covbal$x1_qb10), ")"),
                                      paste0(round(mean(covbal$x1_qb15), 2), " (",
                                             min(covbal$x1_qb15), ", ",
                                             max(covbal$x1_qb15), ")"),
                                      paste0(round(mean(covbal$x1_qb20), 2), " (",
                                             min(covbal$x1_qb20), ", ",
                                             max(covbal$x1_qb20), ")"),
                                      paste0(round(mean(covbal$x1_olr), 2), " (",
                                             min(covbal$x1_olr), ", ",
                                             max(covbal$x1_olr), ")")),
               `Mean (min, max) ` = c(paste0(round(mean(covbal$x2_uw), 2), " (",
                                             min(covbal$x2_uw), ", ",
                                             max(covbal$x2_uw), ")"),
                                      paste0(round(mean(covbal$x2_ols), 2), " (",
                                             min(covbal$x2_ols), ", ",
                                             max(covbal$x2_ols), ")"),
                                      paste0(round(mean(covbal$x2_cbgps), 2), " (",
                                             min(covbal$x2_cbgps), ", ",
                                             max(covbal$x2_cbgps), ")"),
                                      NA,
                                      paste0(round(mean(covbal$x2_qb10), 2), " (",
                                             min(covbal$x2_qb10), ", ",
                                             max(covbal$x2_qb10), ")"),
                                      paste0(round(mean(covbal$x2_qb15), 2), " (",
                                             min(covbal$x2_qb15), ", ",
                                             max(covbal$x2_qb15), ")"),
                                      paste0(round(mean(covbal$x2_qb20), 2), " (",
                                             min(covbal$x2_qb20), ", ",
                                             max(covbal$x2_qb20), ")"),
                                      paste0(round(mean(covbal$x2_olr), 2), " (",
                                             min(covbal$x2_olr), ", ",
                                             max(covbal$x2_olr), ")")),
               `Mean (min, max)  ` = c(paste0(round(mean(covbal$x3_uw), 2), " (",
                                             min(covbal$x3_uw), ", ",
                                             max(covbal$x3_uw), ")"),
                                      paste0(round(mean(covbal$x3_ols), 2), " (",
                                             min(covbal$x3_ols), ", ",
                                             max(covbal$x3_ols), ")"),
                                      paste0(round(mean(covbal$x3_cbgps), 2), " (",
                                             min(covbal$x3_cbgps), ", ",
                                             max(covbal$x3_cbgps), ")"),
                                      NA,
                                      paste0(round(mean(covbal$x3_qb10), 2), " (",
                                             min(covbal$x3_qb10), ", ",
                                             max(covbal$x3_qb10), ")"),
                                      paste0(round(mean(covbal$x3_qb15), 2), " (",
                                             min(covbal$x3_qb15), ", ",
                                             max(covbal$x3_qb15), ")"),
                                      paste0(round(mean(covbal$x3_qb20), 2), " (",
                                             min(covbal$x3_qb20), ", ",
                                             max(covbal$x3_qb20), ")"),
                                      paste0(round(mean(covbal$x3_olr), 2), " (",
                                             min(covbal$x3_olr), ", ",
```

```
                                    max(covbal$x3_olr), ")")) #,
          # `Mean (min, max)   ` = c(paste0(round(mean(covbal$x4_ols), 2), " (",
          #                              min(covbal$x4_ols), ", ",
          #                              max(covbal$x4_ols), ")"),
          #                        paste0(round(mean(covbal$x4_cbgps), 2), " (",
          #                              min(covbal$x4_cbgps), ", ",
          #                              max(covbal$x4_cbgps), ")"),
          #                        NA,
          #                        paste0(round(mean(covbal$x4_qb10), 2), " (",
          #                              min(covbal$x4_qb10), ", ",
          #                              max(covbal$x4_qb10), ")"),
          #                        paste0(round(mean(covbal$x4_qb15), 2), " (",
          #                              min(covbal$x4_qb15), ", ",
          #                              max(covbal$x4_qb15), ")"),
          #                        paste0(round(mean(covbal$x4_qb20), 2), " (",
          #                              min(covbal$x4_qb20), ", ",
          #                              max(covbal$x4_qb20), ")"),
          #                        paste0(round(mean(covbal$x4_olr), 2), " (",
          #                              min(covbal$x4_olr), ", ",
          #                              max(covbal$x4_olr), ")"))
          )
#
# # covariate balance table
# kable(covbal_tab) %>%
#   kable_classic(html_font = "Arial", full_width = FALSE) %>%
#   add_header_above(c("", "X1" = 1, "X2" = 1, "X3" = 1), bold = TRUE) %>%
#   add_header_above(c("Marginally", "Normal" = 1, "Non-Normal" = 1,
#                    "Non-Normal" = 1), bold = TRUE) %>%
#   add_header_above(c("Conditionally", "Normal" = 1, "Normal" = 1,
#                    "Non-Normal" = 1), bold = TRUE)  %>%
#   # add_header_above(c("Expsoure Transformation", "None" = 3,
#   #                    "log(X + 0.001)" = 1), bold = TRUE) %>%
#   add_indent(c(5:7))
```

Table 1 - Inverse Probability Weight and Covariate Balance Distributions [Mean (Min, Max) Version]

```
fin_tab1 <- tibble(Method = c("Unweighted",
                              "Stabilized weight",
                              "Unbalanced covariates",
                          "Ordinary least squares",
                          "Stabilized weight",
                              "Unbalanced covariates",
                          "Covariate balancing generalized propensity score",
                          "Stabilized weight",
                              "Unbalanced covariates",
                          "Quantile binning categories",
                          "10",
                          "Stabilized weight",
                              "Unbalanced covariates",
                          "15",
                          "Stabilized weight",
```

```
                              "Unbalanced covariates",
                          "20",
                          "Stabilized weight",
                              "Unbalanced covariates",
                          "Ordinal logistic regression",
                          "Stabilized weight",
                              "Unbalanced covariates"),
                   `Mean (min, max)` = c(NA, NA, covbal_tab[1, 2],
                                          NA, tab2[1, 2], covbal_tab[2, 2],
                                          NA, tab2[2, 2], covbal_tab[3, 2],
                                          NA,
                                          NA, tab2[4, 2], covbal_tab[5, 2],
                                          NA, tab2[5, 2], covbal_tab[6, 2],
                                          NA, tab2[6, 2], covbal_tab[7, 2],
                                          NA, tab2[7, 2], covbal_tab[8, 2]),
                   `Mean (min, max) ` = c(NA, NA, covbal_tab[1, 3],
                                          NA, tab2[1, 3], covbal_tab[2, 3],
                                          NA, tab2[2, 3], covbal_tab[3, 3],
                                          NA,
                                          NA, tab2[4, 3], covbal_tab[5, 3],
                                          NA, tab2[5, 3], covbal_tab[6, 3],
                                          NA, tab2[6, 3], covbal_tab[7, 3],
                                          NA, tab2[7, 3], covbal_tab[8, 3]),
                   `Mean (min, max)  ` = c(NA, NA, covbal_tab[1, 4],
                                          NA, tab2[1, 4], covbal_tab[2, 4],
                                          NA, tab2[2, 4], covbal_tab[3, 4],
                                          NA,
                                          NA, tab2[4, 4], covbal_tab[5, 4],
                                          NA, tab2[5, 4], covbal_tab[6, 4],
                                          NA, tab2[6, 4], covbal_tab[7, 4],
                                          NA, tab2[7, 4], covbal_tab[8, 4]))

kable(fin_tab1) %>%
  kable_classic(html_font = "Arial", full_width = FALSE) %>%
  add_header_above(c("", "X1" = 1, "X2" = 1, "X3" = 1), bold = TRUE) %>%
  add_header_above(c("Marginally", "Normal" = 1, "Non-Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 1, "Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE)  %>%
  add_indent(c(2:3, 5:6, 8:9, 11:19, 21:22)) %>%
  add_indent(c(2:3, 5:6, 8:9, 12:13, 15:16, 18:19, 21:22))
```

Assessment of Bias

Calculating Bias and Mean Squared Error

```
# will need to calculate all weights using x1 and x2 for each
  # simulated dataset with weighted lrm models
# then calculate bias for exposure coefficient versus truth

# create deciles quantiles for each Austin approach
```

```r
x1_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x1, .x))

x2_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x2, .x))

x3_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x3, .x))

# function to get list of biases via msm approach and Austin approaches
bias_func <- function(data){
  # simulation number
  i <- data$i[1]

  # bias via the Marginal Structural Model Approach

  # generate weighted models
  # unweighted comparison
  x1_uw <- lrm(y1 ~ x1, data = data)
  x2_uw <- lrm(y2 ~ x2, data = data)
  x3_uw <- lrm(y3 ~ x3, data = data)
  #x4_uw  <- lrm(y3 ~ x3, data = data)

  # ols
  x1_ols <- lrm(y1 ~ x1, weights = x1_ols_wts, data = data)
  x2_ols <- lrm(y2 ~ x2, weights = x2_ols_wts, data = data)
  x3_ols <- lrm(y3 ~ x3, weights = x3_ols_wts, data = data)
  #x4_ols <- lrm(y3 ~ x3, weights = x4_ols_wts, data = data)

  # cbgps
  x1_cbgps <- lrm(y1 ~ x1, weights = x1_cbgps_wts, data = data)
  x2_cbgps <- lrm(y2 ~ x2, weights = x2_cbgps_wts, data = data)
  x3_cbgps <- lrm(y3 ~ x3, weights = x3_cbgps_wts, data = data)
  #x4_cbgps <- lrm(y3 ~ x3, weights = x4_cbgps_wts, data = data)

  # qb10
  x1_qb10 <- lrm(y1 ~ x1, weights = x1_qb10_wts, data = data)
  x2_qb10 <- lrm(y2 ~ x2, weights = x2_qb10_wts, data = data)
  x3_qb10 <- lrm(y3 ~ x3, weights = x3_qb10_wts, data = data)
  #x4_qb10 <- lrm(y3 ~ x3, weights = x4_qb10_wts, data = data)

  # qb15
  x1_qb15 <- lrm(y1 ~ x1, weights = x1_qb15_wts, data = data)
  x2_qb15 <- lrm(y2 ~ x2, weights = x2_qb15_wts, data = data)
  x3_qb15 <- lrm(y3 ~ x3, weights = x3_qb15_wts, data = data)
  #x4_qb15 <- lrm(y3 ~ x3, weights = x4_qb15_wts, data = data)

  # qb20
  x1_qb20 <- lrm(y1 ~ x1, weights = x1_qb20_wts, data = data)
  x2_qb20 <- lrm(y2 ~ x2, weights = x2_qb20_wts, data = data)
  x3_qb20 <- lrm(y3 ~ x3, weights = x3_qb20_wts, data = data)
  #x4_qb20 <- lrm(y3 ~ x3, weights = x4_qb20_wts, data = data)

  # olr
  x1_olr <- lrm(y1 ~ x1, weights = x1_olr_wts, data = data)
  x2_olr <- lrm(y2 ~ x2, weights = x2_olr_wts, data = data)
```

```r
x3_olr <- lrm(y3 ~ x3, weights = x3_olr_wts, data = data)
#x4_olr <- lrm(y3 ~ x3, weights = x4_olr_wts, data = data)

# bias via the Marginal Structural Model Approach

# unweighted bias
x1_uw_bias <- true_x1 - x1_uw$coefficient[2]
x2_uw_bias <- true_x2 - x2_uw$coefficient[2]
x3_uw_bias <- true_x3 - x3_uw$coefficient[2]
#x4_uw_bias <- true_x3 - x4_ols_uw$coefficient[2]

# ols
x1_ols_bias <- true_x1 - x1_ols$coefficient[2]
x2_ols_bias <- true_x2 - x2_ols$coefficient[2]
x3_ols_bias <- true_x3 - x3_ols$coefficient[2]
#x4_ols_bias <- true_x3 - x4_ols$coefficient[2]

# cbgps
x1_cbgps_bias <- true_x1 - x1_cbgps$coefficient[2]
x2_cbgps_bias <- true_x2 - x2_cbgps$coefficient[2]
x3_cbgps_bias <- true_x3 - x3_cbgps$coefficient[2]
#x4_cbgps_bias <- true_x3 - x4_cbgps$coefficient[2]

# qb10
x1_qb10_bias <- true_x1 - x1_qb10$coefficient[2]
x2_qb10_bias <- true_x2 - x2_qb10$coefficient[2]
x3_qb10_bias <- true_x3 - x3_qb10$coefficient[2]
#x4_qb10_bias <- true_x3 - x4_qb10$coefficient[2]

# qb15
x1_qb15_bias <- true_x1 - x1_qb15$coefficient[2]
x2_qb15_bias <- true_x2 - x2_qb15$coefficient[2]
x3_qb15_bias <- true_x3 - x3_qb15$coefficient[2]
#x4_qb15_bias <- true_x3 - x4_qb15$coefficient[2]

# qb20
x1_qb20_bias <- true_x1 - x1_qb20$coefficient[2]
x2_qb20_bias <- true_x2 - x2_qb20$coefficient[2]
x3_qb20_bias <- true_x3 - x3_qb20$coefficient[2]
#x4_qb20_bias <- true_x3 - x4_qb20$coefficient[2]

# olr
x1_olr_bias <- true_x1 - x1_olr$coefficient[2]
x2_olr_bias <- true_x2 - x2_olr$coefficient[2]
x3_olr_bias <- true_x3 - x3_olr$coefficient[2]
#x4_olr_bias <- true_x3 - x4_olr$coefficient[2]

# bias via the Austin, 2018 approach

# first have to generate probability of having each exposure decile in each model

# unweighted
x1_uw_qs <- map_dbl(x1_quants,
```

```
                 ~ predict(x1_uw,
                           newdata = .x,
                           type = "fitted"))
x2_uw_qs <- map_dbl(x2_quants,
                 ~ predict(x2_uw,
                           newdata = .x,
                           type = "fitted"))
x3_uw_qs <- map_dbl(x3_quants,
                 ~ predict(x3_uw,
                           newdata = .x,
                           type = "fitted"))
# x4_uw_qs <- map_dbl(x3_quants,
#                  ~ predict(x4_uw, # only change here because only difference in weights, not exposu
#                            newdata = .x,
#                            type = "fitted"))

# x1
x1_uw_bias2 <- true2_x1_qs - x1_uw_qs

# x2
x2_uw_bias2 <- true2_x2_qs - x2_uw_qs

# x3
x3_uw_bias2 <- true2_x3_qs - x3_uw_qs

# x4
# x4_uw_bias2 <- true2_x3_qs - x4_uw_qs # truth is the same as x3 with different weighting

# ols
x1_ols_qs <- map_dbl(x1_quants,
                 ~ predict(x1_ols,
                           newdata = .x,
                           type = "fitted"))
x2_ols_qs <- map_dbl(x2_quants,
                 ~ predict(x2_ols,
                           newdata = .x,
                           type = "fitted"))
x3_ols_qs <- map_dbl(x3_quants,
                 ~ predict(x3_ols,
                           newdata = .x,
                           type = "fitted"))
# x4_ols_qs <- map_dbl(x3_quants,
#                  ~ predict(x4_ols, # only change here because only difference in weights, not expos
#                            newdata = .x,
#                            type = "fitted"))

# x1
x1_ols_bias2 <- true2_x1_qs - x1_ols_qs

# x2
x2_ols_bias2 <- true2_x2_qs - x2_ols_qs

# x3
```

```r
x3_ols_bias2 <- true2_x3_qs - x3_ols_qs

# x4
# x4_ols_bias2 <- true2_x3_qs - x4_ols_qs # truth is the same as x3 with different weighting

# cbgps
x1_cbgps_qs <- map_dbl(x1_quants,
                 ~ predict(x1_cbgps,
                           newdata = .x,
                           type = "fitted"))
x2_cbgps_qs <- map_dbl(x2_quants,
                 ~ predict(x2_cbgps,
                           newdata = .x,
                           type = "fitted"))
x3_cbgps_qs <- map_dbl(x3_quants,
                 ~ predict(x3_cbgps,
                           newdata = .x,
                           type = "fitted"))
# x4_cbgps_qs <- map_dbl(x3_quants,
#                  ~ predict(x4_cbgps, # only change here because only difference in weights, not exp
#                            newdata = .x,
#                            type = "fitted"))

# x1
x1_cbgps_bias2 <- true2_x1_qs - x1_cbgps_qs

# x2
x2_cbgps_bias2 <- true2_x2_qs - x2_cbgps_qs

# x3
x3_cbgps_bias2 <- true2_x3_qs - x3_cbgps_qs

# x4
#x4_cbgps_bias2 <- true2_x3_qs - x4_cbgps_qs # truth is the same as x3 with different weighting

# qb10
x1_qb10_qs <- map_dbl(x1_quants,
                 ~ predict(x1_qb10,
                           newdata = .x,
                           type = "fitted"))
x2_qb10_qs <- map_dbl(x2_quants,
                 ~ predict(x2_qb10,
                           newdata = .x,
                           type = "fitted"))
x3_qb10_qs <- map_dbl(x3_quants,
                 ~ predict(x3_qb10,
                           newdata = .x,
                           type = "fitted"))
# x4_qb10_qs <- map_dbl(x3_quants,
#                  ~ predict(x4_qb10, # only change here because only difference in weights, not expo
#                            newdata = .x,
#                            type = "fitted"))
```

```
# x1
x1_qb10_bias2 <- true2_x1_qs - x1_qb10_qs

# x2
x2_qb10_bias2 <- true2_x2_qs - x2_qb10_qs

# x3
x3_qb10_bias2 <- true2_x3_qs - x3_qb10_qs

# x4
#x4_qb10_bias2 <- true2_x3_qs - x4_qb10_qs # truth is the same as x3 with different weighting

# qb15
x1_qb15_qs <- map_dbl(x1_quants,
                 ~ predict(x1_qb15,
                           newdata = .x,
                           type = "fitted"))
x2_qb15_qs <- map_dbl(x2_quants,
                 ~ predict(x2_qb15,
                           newdata = .x,
                           type = "fitted"))
x3_qb15_qs <- map_dbl(x3_quants,
                 ~ predict(x3_qb15,
                           newdata = .x,
                           type = "fitted"))
# x4_qb15_qs <- map_dbl(x3_quants,
#                  ~ predict(x4_qb15, # only change here because only difference in weights, not expo
#                            newdata = .x,
#                            type = "fitted"))

# x1
x1_qb15_bias2 <- true2_x1_qs - x1_qb15_qs

# x2
x2_qb15_bias2 <- true2_x2_qs - x2_qb15_qs

# x3
x3_qb15_bias2 <- true2_x3_qs - x3_qb15_qs

# x4
#x4_qb15_bias2 <- true2_x3_qs - x4_qb15_qs # truth is the same as x3 with different weighting

# qb20
x1_qb20_qs <- map_dbl(x1_quants,
                 ~ predict(x1_qb20,
                           newdata = .x,
                           type = "fitted"))
x2_qb20_qs <- map_dbl(x2_quants,
                 ~ predict(x2_qb20,
                           newdata = .x,
                           type = "fitted"))
x3_qb20_qs <- map_dbl(x3_quants,
                 ~ predict(x3_qb20,
```

```
                             newdata = .x,
                             type = "fitted"))
# x4_qb20_qs <- map_dbl(x3_quants,
#                   ~ predict(x4_qb20, # only change here because only difference in weights, not expo
#                             newdata = .x,
#                             type = "fitted"))

# x1
x1_qb20_bias2 <- true2_x1_qs - x1_qb20_qs

# x2
x2_qb20_bias2 <- true2_x2_qs - x2_qb20_qs

# x3
x3_qb20_bias2 <- true2_x3_qs - x3_qb20_qs

# x4
#x4_qb20_bias2 <- true2_x3_qs - x4_qb20_qs # truth is the same as x3 with different weighting

# olr
x1_olr_qs <- map_dbl(x1_quants,
                 ~ predict(x1_olr,
                           newdata = .x,
                           type = "fitted"))
x2_olr_qs <- map_dbl(x2_quants,
                 ~ predict(x2_olr,
                           newdata = .x,
                           type = "fitted"))
x3_olr_qs <- map_dbl(x3_quants,
                 ~ predict(x3_olr,
                           newdata = .x,
                           type = "fitted"))
# x4_olr_qs <- map_dbl(x3_quants,
#                   ~ predict(x4_olr, # only change here because only difference in weights, not expos
#                             newdata = .x,
#                             type = "fitted"))

# x1
x1_olr_bias2 <- true2_x1_qs - x1_olr_qs

# x2
x2_olr_bias2 <- true2_x2_qs - x2_olr_qs

# x3
x3_olr_bias2 <- true2_x3_qs - x3_olr_qs

# x4
#x4_olr_bias2 <- true2_x3_qs - x4_olr_qs # truth is the same as x3 with different weighting


# output dataframe
bias1 <- data.frame(i,
           x1_uw_bias, x2_uw_bias, x3_uw_bias,
```

```
              x1_ols_bias, x2_ols_bias, x3_ols_bias,
              x1_cbgps_bias, x2_cbgps_bias, x3_cbgps_bias,
              x1_qb10_bias, x2_qb10_bias, x3_qb10_bias,
              x1_qb15_bias, x2_qb15_bias, x3_qb15_bias,
              x1_qb20_bias, x2_qb20_bias, x3_qb20_bias,
              x1_olr_bias, x2_olr_bias, x3_olr_bias)

  bias2 <- data.frame(quantile = c(1:9),
              x1_uw_bias2, x2_uw_bias2, x3_uw_bias2,
              x1_ols_bias2, x2_ols_bias2, x3_ols_bias2,
              x1_cbgps_bias2, x2_cbgps_bias2, x3_cbgps_bias2,
              x1_qb10_bias2, x2_qb10_bias2, x3_qb10_bias2,
              x1_qb15_bias2, x2_qb15_bias2, x3_qb15_bias2,
              x1_qb20_bias2, x2_qb20_bias2, x3_qb20_bias2,
              x1_olr_bias2, x2_olr_bias2, x3_olr_bias2) %>%
    pivot_wider(names_from = quantile,
                values_from = x1_uw_bias2:x3_olr_bias2)

  data.frame(bias1, bias2)
}

# # get bias across all simulations
# bias <- map_df(sims, ~ bias_func(.x))

# run in parallel with furrr
plan(multisession, workers = 7)
bias <- future_map_dfr(sims, ~ bias_func(.x))
Save(bias)
```

Decile Approach

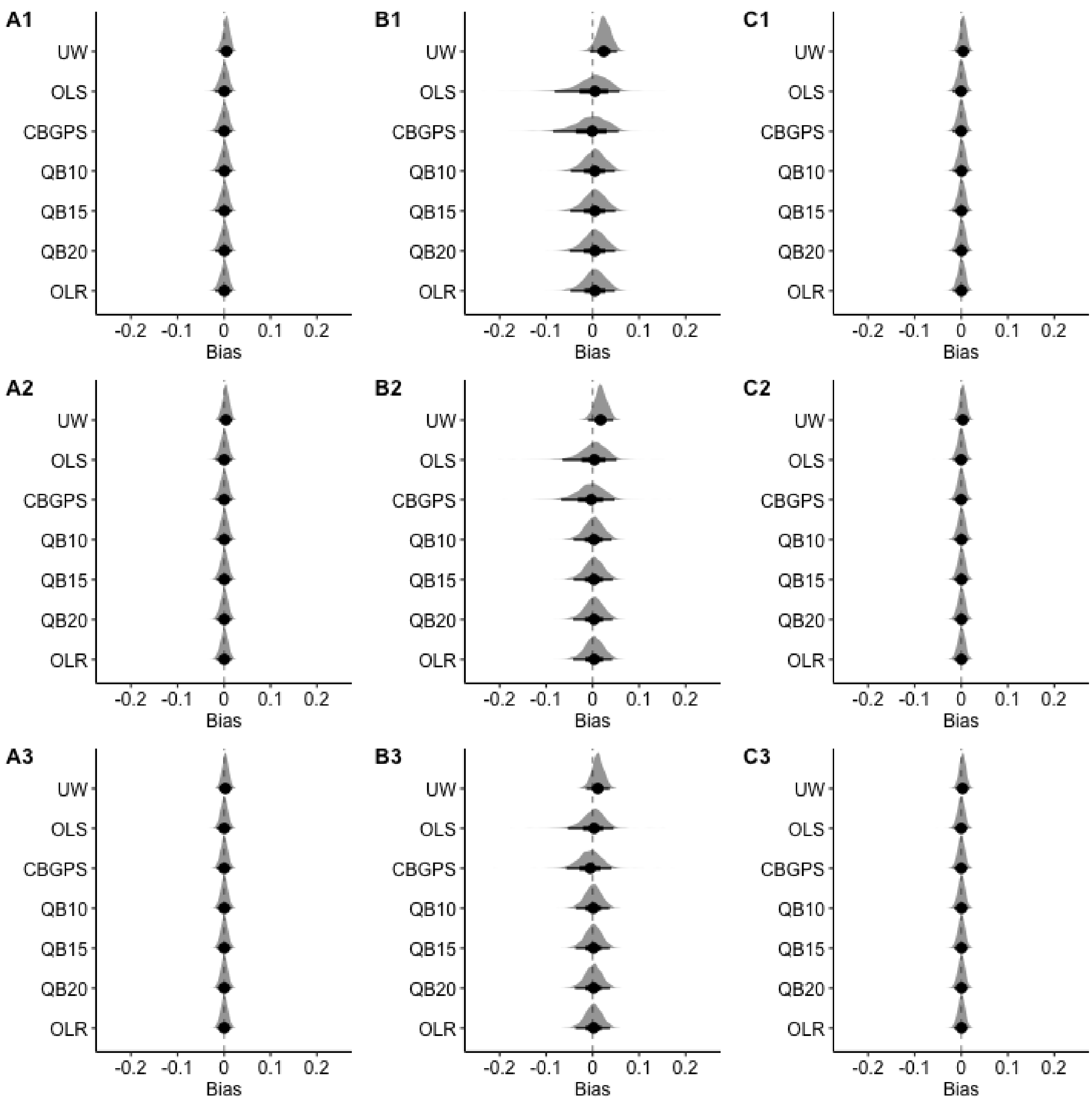
Supplemental Figure 1.5 - Bias at Deciles

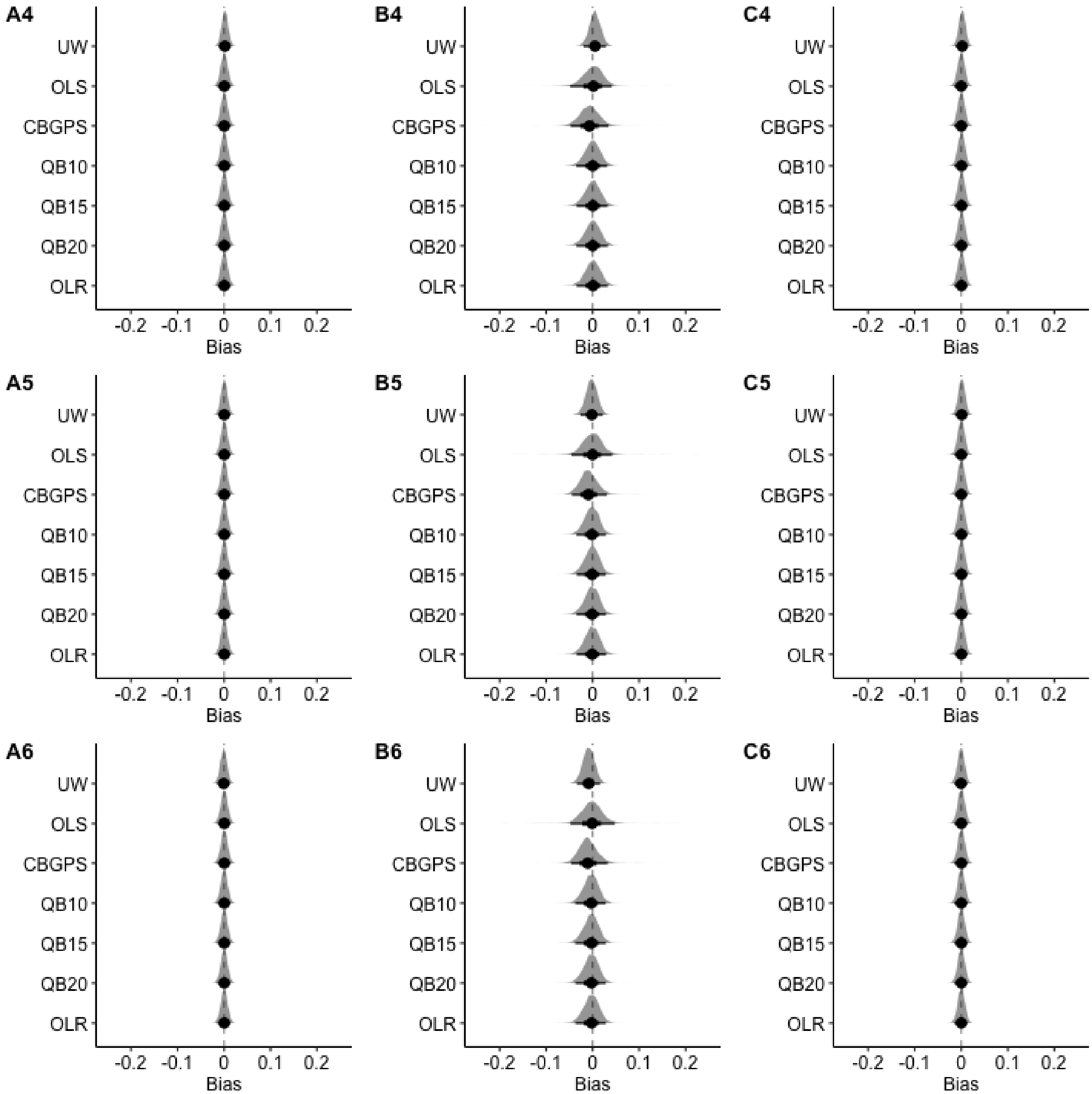
```
Load(bias)
# function to plot austin bias plots
aus_bias_plot <- function(decile) {
  # make bias dataframes
  x1_bias2 <- bias %>%
    select(starts_with("x1") & contains("bias2") & ends_with(as.character(decile))) %>%
    gather(label, bias) %>%
    mutate(label = factor(label,
                          levels = c(paste0("x1_uw_bias2_", decile),
                                     paste0("x1_ols_bias2_", decile),
                                     paste0("x1_cbgps_bias2_", decile),
                                     paste0("x1_qb10_bias2_", decile),
                                     paste0("x1_qb15_bias2_", decile),
                                     paste0("x1_qb20_bias2_", decile),
                                     paste0("x1_olr_bias2_", decile))))

  x2_bias2 <- bias %>%
    select(starts_with("x2") & contains("bias2") & ends_with(as.character(decile))) %>%
    gather(label, bias) %>%
```

```
    mutate(label = factor(label,
                          levels = c(paste0("x2_uw_bias2_", decile),
                                     paste0("x2_ols_bias2_", decile),
                                     paste0("x2_cbgps_bias2_", decile),
                                     paste0("x2_qb10_bias2_", decile),
                                     paste0("x2_qb15_bias2_", decile),
                                     paste0("x2_qb20_bias2_", decile),
                                     paste0("x2_olr_bias2_", decile))))

  x3_bias2 <- bias %>%
    select(starts_with("x3") & contains("bias2") & ends_with(as.character(decile))) %>%
    gather(label, bias) %>%
    mutate(label = factor(label,
                          levels = c(paste0("x3_uw_bias2_", decile),
                                     paste0("x3_ols_bias2_", decile),
                                     paste0("x3_cbgps_bias2_", decile),
                                     paste0("x3_qb10_bias2_", decile),
                                     paste0("x3_qb15_bias2_", decile),
                                     paste0("x3_qb20_bias2_", decile),
                                     paste0("x3_olr_bias2_", decile))))

  # make plots
  x1_bias_plot2 <- ggplot(x1_bias2, aes(y = fct_rev(label), x = bias)) +
    stat_halfeye() +
    geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
    scale_y_discrete(name = "", labels = c("OLR", "QB20", "QB15", "QB10", "CBGPS", "OLS", "UW")) +
    scale_x_continuous(name = "Bias", limits = c(-0.25, 0.25),
                       breaks = seq(-0.2, 0.2, 0.1), labels = seq(-0.2, 0.2, 0.1)) +
    theme

  x2_bias_plot2 <- ggplot(x2_bias2, aes(y = fct_rev(label), x = bias)) +
    stat_halfeye() +
    geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
    scale_y_discrete(name = "", labels = c("OLR", "QB20", "QB15", "QB10", "CBGPS", "OLS", "UW")) +
    scale_x_continuous(name = "Bias", limits = c(-0.25, 0.25),
                       breaks = seq(-0.2, 0.2, 0.1), labels = seq(-0.2, 0.2, 0.1)) +
    theme

  x3_bias_plot2 <- ggplot(x3_bias2, aes(y = fct_rev(label), x = bias)) +
    stat_halfeye() +
    geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
    scale_y_discrete(name = "", labels = c("OLR", "QB20", "QB15", "QB10", "CBGPS", "OLS", "UW")) +
    scale_x_continuous(name = "Bias", limits = c(-0.25, 0.25),
                       breaks = seq(-0.2, 0.2, 0.1), labels = seq(-0.2, 0.2, 0.1)) +
    theme

# combine plots
  ggarrange(x1_bias_plot2, x2_bias_plot2, x3_bias_plot2,
            labels = c(paste0("A", decile), paste0("B", decile), paste0("C", decile)),
            nrow = 1)
}

# plot all deciles
```

```r
aus_plots <- map(c(1:9), ~ aus_bias_plot(.x))
```

```r
# plots in chunks
ggarrange(aus_plots[[1]], aus_plots[[2]], aus_plots[[3]], ncol = 1)
ggarrange(aus_plots[[4]], aus_plots[[5]], aus_plots[[6]], ncol = 1)
ggarrange(aus_plots[[7]], aus_plots[[8]], aus_plots[[9]], ncol = 1)
```




A4
B4
C4
A5
B5
C5
A6
B6
C6
UW
OLS
CBGPS
QB10
QB15
QB20
OLR
-0.2
-0.1
0
0.1
0.2
Bias

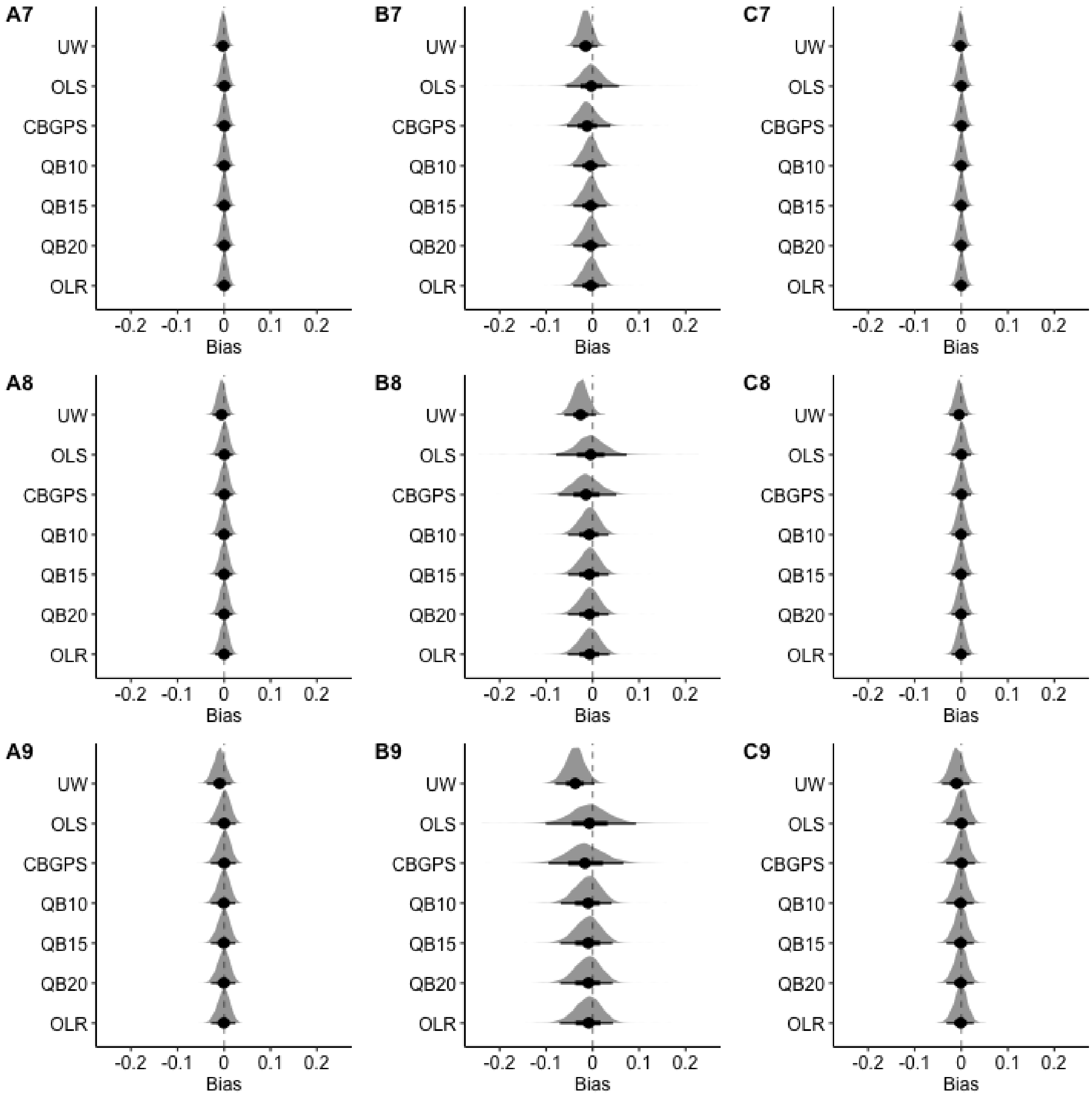


Figure legend: Column A) Homoscedastic Exposure, Conditionally Normal, Marginally Normal, Column B) Homoscedastic Exposure, Conditionally Normal, Marginally Non-Normal, Column C) Heteroscedastic Exposure, Conditionally Non-Normal, Marginally Non-Normal. The number next to each panel represents the decile, such that "A1" is the 1st decile of the Homoscedastic, Conditionally Normal, Marginally Normal Exposure.

Figure 2 - Mean Squared Error at Deciles

```
# calculate mse for each scenario
austin_mse <- expand_grid(Decile = c(1:9),
                          Method = c("UW",
                                     "OLS",
                                     "CBGPS",
```

```r
                                     "QB10",
                                     "QB15",
                                     "QB20",
                                     "OLR"),
                          Exposure = c("x1",
                                       "x2",
                                       "x3"),
                          MSE = NA)

# now loop through and fill in cells
for(i in 1:nrow(austin_mse)){
  # create column name of bias tibble
  col_name <- paste0(austin_mse$Exposure[i], "_",
                     tolower(austin_mse$Method[i]), "_bias2_",
                     austin_mse$Decile[i])
  austin_mse$MSE[i] <- mean(bias[[col_name]]^2)
}

# now make plot

# first update levels
austin_mse <- austin_mse %>%
  mutate(Decile = factor(Decile),
         Method = factor(Method, levels = c("UW",
                                            "QB10",
                                     "OLS",
                                     "QB15",
                                     "CBGPS",
                                     "QB20",
                                     "OLR")),
         Exposure = factor(Exposure, levels = c("x1",
                                       "x2",
                                       "x3"))) %>%
  mutate(name = factor(Exposure,
                       labels = c(expression(paste("A) ", X[1])),
                                  expression(paste("B) ", X[2])),
                                  expression(paste("C) ", X[3])))))

# now create dataframe with vertical horizontal lines by facets (from MSM mse)
# msm_mse <- tab3 %>% filter(!is.na(X1)) %>%
#   mutate(Method = levels(austin_mse$Method)) %>%
#   pivot_longer(cols = X1:X4, names_to = "Exposure", values_to = "MSE") %>%
#   mutate(Exposure = tolower(Exposure)) %>%
#   mutate(Method = factor(Method, levels = c("OLS",
#                                       "CBGPS",
#                                       "QB10",
#                                       "QB15",
#                                       "QB20",
#                                       "OLR")),
#          Exposure = factor(Exposure, levels = c("x1",
#                                         "x2",
#                                         "x3",
#                                         "x4"))) %>%
```

```r
#   mutate(name = factor(Exposure,
#                        labels = c(expression(paste("A. ", X[1], " - Homoscedastic, Naimi")),
#                                   expression(paste("B. ", X[2], " - Homoscedastic, Skewed")),
#                                   expression(paste("C. ", X[3], " - Heterocedastic, Naimi")),
#                                   expression(paste("D. ", X[4], " - Heterocedastic, log(Naimi + 0.001

# # make reference tibble to jitter points
# point_jitter <- tibble(Method = levels(austin_mse$Method),
#                        Jitter = round(seq(-0.15, 0.15, 0.05), 2))
#
# # add to austin_mse
# austin_mse <- left_join(austin_mse, point_jitter, by = "Method") %>%
#   mutate(dec = as.numeric(Decile) + Jitter,
#          Method = factor(Method, levels = c("UW",
#                                             "OLS",
#                                             "CBGPS",
#                                             "QB10",
#                                             "QB15",
#                                             "QB20",
#                                             "OLR")))

# now plot
austin_mse %>%
  ggplot(aes(x = MSE, y = Decile)) +
  # geom_vline(data = msm_mse,
  #            aes(xintercept = MSE, color = Method),
  #            linetype = "longdash", alpha = 0.9) +
  geom_point(aes(shape = Method, color = Method), size = 1) +
  facet_wrap(~name, ncol = 1, labeller = "label_parsed", scales = "free_x") +
  scale_color_grey(guide = guide_legend(nrow = 2)) +
  scale_shape_manual(values = 1:nlevels(austin_mse$Method)) +
  #scale_y_continuous(name = "Decile", breaks = 1:9) + # if you'd like to jitter things
  theme +
  theme(panel.grid.major.y = element_line(),
        axis.text = element_text(size = 8),
        axis.title = element_text(size = 9),
        legend.title = element_text(size = 9),
        legend.text = element_text(size = 8),
        strip.text = element_text(hjust = 0, size = 10),
        strip.background = element_blank())

# save plot
ggsave("figs/fig2.pdf", width = 4, height = 6)
# embed the font
embed_fonts("figs/fig2.pdf")
```

Marginal Log Odds Ratio Approach

Figure 3 - Marginal Log Odds Ratio Approach Bias

```r
# make df of x1:4 biases
x1_bias <- bias %>%
```

```r
  select(starts_with("x1") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
                        levels = c("x1_uw_bias",
                                   "x1_ols_bias",
                                   "x1_cbgps_bias",
                                   "x1_qb10_bias",
                                   "x1_qb15_bias",
                                   "x1_qb20_bias",
                                   "x1_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

x2_bias <- bias %>%
  select(starts_with("x2") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
                        levels = c("x2_uw_bias",
                                   "x2_ols_bias",
                                   "x2_cbgps_bias",
                                   "x2_qb10_bias",
                                   "x2_qb15_bias",
                                   "x2_qb20_bias",
                                   "x2_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

x3_bias <- bias %>%
  select(starts_with("x3") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
                        levels = c("x3_uw_bias",
                                   "x3_ols_bias",
                                   "x3_cbgps_bias",
                                   "x3_qb10_bias",
                                   "x3_qb15_bias",
                                   "x3_qb20_bias",
                                   "x3_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

# combine into one dataset so can make facet plot
mse_bias_plot <- bind_rows(x1_bias, x2_bias, x3_bias) %>%
  mutate(facet = factor(facet, levels = c("x1", "x2", "x3")),
         lab = factor(lab,levels = c("uw_bias",
                                   "ols_bias",
                                   "cbgps_bias",
                                   "qb10_bias",
                                   "qb15_bias",
                                   "qb20_bias",
                                   "olr_bias"))) %>%
  mutate(name = factor(facet,
                       labels = c(expression(paste("A) ", X[1])),
```

```
                                  expression(paste("B) ", X[2])),
                                  expression(paste("C) ", X[3])))))

# plot
mse_bias_plot %>%
  ggplot(aes(y = fct_rev(lab), x = bias)) +
  stat_halfeye() +
  geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
  facet_wrap(~name, ncol = 1, labeller = "label_parsed", scales = "free_x") +
  scale_y_discrete(name = "", labels = c("OLR", "QB20", "QB15", "QB10", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.25, 0.25),
                     breaks = seq(-0.2, 0.2, 0.1), labels = seq(-0.2, 0.2, 0.1)) +
  theme +
  theme(panel.grid.major.y = element_line(),
        axis.text = element_text(size = 8),
        axis.title = element_text(size = 9),
        legend.title = element_text(size = 9),
        legend.text = element_text(size = 8),
        strip.text = element_text(hjust = 0, size = 10),
        strip.background = element_blank())


# save plot
ggsave("figs/fig3.pdf", width = 4, height = 6)
# embed the font
embed_fonts("figs/fig3.pdf")
```

Mean Squared Error

Table 2 - Marginal Log Odds Ratio Approach Mean Squared Error

```
# now make table of mean squared error, which is the mean of the squared biases (or errors)
tab3 <- tibble(Method = c("Unweighted",
                          "Ordinary least squares",
                          "Covariate balancing generalized propensity score",
                          "Quantile binning categories",
                          "10",
                          "15",
                          "20",
                          "Ordinal logistic regression"),
               `X1` = c(mean(bias$x1_uw_bias^2),
                        mean(bias$x1_ols_bias^2),
                        mean(bias$x1_cbgps_bias^2),
                        NA,
                        mean(bias$x1_qb10_bias^2),
                        mean(bias$x1_qb15_bias^2),
                        mean(bias$x1_qb20_bias^2),
                        mean(bias$x1_olr_bias^2)),
               `X2` = c(mean(bias$x2_uw_bias^2),
                        mean(bias$x2_ols_bias^2),
                        mean(bias$x2_cbgps_bias^2),
                        NA,
```

```r
                         mean(bias$x2_qb10_bias^2),
                         mean(bias$x2_qb15_bias^2),
                         mean(bias$x2_qb20_bias^2),
                         mean(bias$x2_olr_bias^2)),
                `X3` = c(mean(bias$x3_uw_bias^2),
                         mean(bias$x3_ols_bias^2),
                         mean(bias$x3_cbgps_bias^2),
                         NA,
                         mean(bias$x3_qb10_bias^2),
                         mean(bias$x3_qb15_bias^2),
                         mean(bias$x3_qb20_bias^2),
                         mean(bias$x3_olr_bias^2))#,
                # `X4` = c(mean(bias$x4_uw_bias^2),
                 #        mean(bias$x4_ols_bias^2),
                #                    mean(bias$x4_cbgps_bias^2),
                #                    NA,
                #                    mean(bias$x4_qb10_bias^2),
                #                    mean(bias$x4_qb15_bias^2),
                #                    mean(bias$x4_qb20_bias^2),
                #                    mean(bias$x4_olr_bias^2))
                )

# make table
kable(tab3, digits = 4) %>%
  kable_classic(html_font = "Arial", full_width = FALSE) %>%
  add_header_above(c("Marginally", "Normal" = 1, "Non-Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 1, "Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE) %>%
  add_indent(c(5:7))
```

Supplemental Table 1.4 - Marginal Log Odds Ratio Approach Bias

```r
# now make table of biases
tab4 <- tibble(Method = c("Unweighted",
                          "Ordinary least squares",
                          "Covariate balancing generalized propensity score",
                          "Quantile binning categories",
                          "10",
                          "15",
                          "20",
                          "Ordinal logistic regression"),
               `Median Bias (IQR)` = c(paste0(round(median(bias$x1_uw_bias), 4), " (",
                                        round(quantile(bias$x1_uw_bias, 0.25), 3), ", ",
                                        round(quantile(bias$x1_uw_bias, 0.75), 3), ")"),
                                   paste0(round(median(bias$x1_ols_bias), 4), " (",
                                        round(quantile(bias$x1_ols_bias, 0.25), 3), ", ",
                                        round(quantile(bias$x1_ols_bias, 0.75), 3), ")"),
                                 paste0(round(median(bias$x1_cbgps_bias), 4), " (",
                                        round(quantile(bias$x1_cbgps_bias, 0.25), 3), ", ",
                                        round(quantile(bias$x1_cbgps_bias, 0.75), 3), ")"),
                                 NA,
```

```r
                                         paste0(round(median(bias$x1_qb10_bias), 4), " (",
                                                round(quantile(bias$x1_qb10_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x1_qb10_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x1_qb15_bias), 4), " (",
                                                round(quantile(bias$x1_qb15_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x1_qb15_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x1_qb20_bias), 4), " (",
                                                round(quantile(bias$x1_qb20_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x1_qb20_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x1_olr_bias), 4), " (",
                                                round(quantile(bias$x1_olr_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x1_olr_bias, 0.75), 3), ")")),
                     `Median Bias (IQR) ` = c(paste0(round(median(bias$x2_uw_bias), 4), " (",
                                                round(quantile(bias$x2_uw_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_uw_bias, 0.75), 3), ")"),
                                            paste0(round(median(bias$x2_ols_bias), 4), " (",
                                                round(quantile(bias$x2_ols_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_ols_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x2_cbgps_bias), 4), " (",
                                                round(quantile(bias$x2_cbgps_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_cbgps_bias, 0.75), 3), ")"),
                                         NA,
                                         paste0(round(median(bias$x2_qb10_bias), 4), " (",
                                                round(quantile(bias$x2_qb10_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_qb10_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x2_qb15_bias), 4), " (",
                                                round(quantile(bias$x2_qb15_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_qb15_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x2_qb20_bias), 4), " (",
                                                round(quantile(bias$x2_qb20_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_qb20_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x2_olr_bias), 4), " (",
                                                round(quantile(bias$x2_olr_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x2_olr_bias, 0.75), 3), ")")),
                     `Median Bias (IQR)  ` = c(paste0(round(median(bias$x3_uw_bias), 4), " (",
                                                round(quantile(bias$x3_uw_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_uw_bias, 0.75), 3), ")"),
                                             paste0(round(median(bias$x3_ols_bias), 4), " (",
                                                round(quantile(bias$x3_ols_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_ols_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x3_cbgps_bias), 4), " (",
                                                round(quantile(bias$x3_cbgps_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_cbgps_bias, 0.75), 3), ")"),
                                         NA,
                                         paste0(round(median(bias$x3_qb10_bias), 4), " (",
                                                round(quantile(bias$x3_qb10_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_qb10_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x3_qb15_bias), 4), " (",
                                                round(quantile(bias$x3_qb15_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_qb15_bias, 0.75), 3), ")"),
                                         paste0(round(median(bias$x3_qb20_bias), 4), " (",
                                                round(quantile(bias$x3_qb20_bias, 0.25), 3), ", ",
                                                round(quantile(bias$x3_qb20_bias, 0.75), 3), ")"),
```

```r
                                paste0(round(median(bias$x3_olr_bias), 4), " (",
                                       round(quantile(bias$x3_olr_bias, 0.25), 3), ", ",
                                       round(quantile(bias$x3_olr_bias, 0.75), 3), ")"))#,
               # `Median Bias (IQR)   ` = c(paste0(round(median(bias$x4_uw_bias), 4), " (",
               #                                 round(quantile(bias$x4_uw_bias, 0.25), 3), ", ",
               #                                 round(quantile(bias$x4_uw_bias, 0.75), 3), ")"),
               #                                 paste0(round(median(bias$x4_ols_bias), 4), " (",
               #                                  round(quantile(bias$x4_ols_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_ols_bias, 0.75), 3), ")"),
               #                          paste0(round(median(bias$x4_cbgps_bias), 4), " (",
               #                                  round(quantile(bias$x4_cbgps_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_cbgps_bias, 0.75), 3), ")"),
               #                          NA,
               #                          paste0(round(median(bias$x4_qb10_bias), 4), " (",
               #                                  round(quantile(bias$x4_qb10_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_qb10_bias, 0.75), 3), ")"),
               #                          paste0(round(median(bias$x4_qb15_bias), 4), " (",
               #                                  round(quantile(bias$x4_qb15_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_qb15_bias, 0.75), 3), ")"),
               #                          paste0(round(median(bias$x4_qb20_bias), 4), " (",
               #                                  round(quantile(bias$x4_qb20_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_qb20_bias, 0.75), 3), ")"),
               #                          paste0(round(median(bias$x4_olr_bias), 4), " (",
               #                                  round(quantile(bias$x4_olr_bias, 0.25), 3), ", ",
               #                                  round(quantile(bias$x4_olr_bias, 0.75), 3), ")"))
               )

# make table
kable(tab4) %>%
  kable_classic(html_font = "Arial", full_width = TRUE) %>%
  add_header_above(c("", "X1" = 1, "X2" = 1, "X3" = 1), bold = TRUE) %>%
  add_header_above(c("Marginally", "Normal" = 1, "Non-Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 1, "Normal" = 1,
                     "Non-Normal" = 1), bold = TRUE) %>%
  # add_header_above(c("Expsoure Transformation", "None" = 3,
  #                    "log(X + 0.001)" = 1), bold = TRUE) %>%
  add_indent(c(5:7))
```

Conditionally	Normal	Normal	Non-Normal
Marginally	Normal	Non-Normal	Non-Normal
	X1	X2	X3
Method	Median Bias (IQR)	Median Bias (IQR)	Median Bias (IQR)
Unweighted	-0.0587 (-0.105, -0.014)	-0.072 (-0.096, -0.05)	-0.0413 (-0.076, -0.008)
Ordinary least squares	-0.0054 (-0.056, 0.049)	-0.0132 (-0.063, 0.035)	6e-04 (-0.037, 0.039)
Covariate balancing generalized propensity score	-0.0042 (-0.056, 0.05)	-0.0145 (-0.065, 0.037)	5e-04 (-0.037, 0.038)
Quantile binning categories			
10	-0.0101 (-0.058, 0.043)	-0.0161 (-0.051, 0.019)	-0.0072 (-0.044, 0.029)
15	-0.0091 (-0.057, 0.043)	-0.0165 (-0.051, 0.018)	-0.0066 (-0.043, 0.029)
20	-0.0091 (-0.057, 0.043)	-0.0168 (-0.051, 0.019)	-0.0061 (-0.043, 0.03)
Ordinal logistic regression	-0.0086 (-0.057, 0.044)	-0.0158 (-0.051, 0.018)	-0.0058 (-0.043, 0.03)

Session Info

```
sessionInfo()
```

```
R version 4.1.0 (2021-05-18)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS 12.4

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapack.dylib

Random number generation:
 RNG:     L'Ecuyer-CMRG
 Normal:  Inversion
 Sample:  Rejection

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] ggdist_3.0.0      extrafont_0.17    doParallel_1.0.16 iterators_1.0.14
 [5] foreach_1.5.2     furrr_0.2.3       future_1.23.0     ggpubr_0.4.0
 [9] kableExtra_1.3.4  lme4_1.1-27.1     Matrix_1.3-4      cobalt_4.3.1
[13] MatchThem_1.0.1   WeightIt_0.12.0   mice_3.13.0       rms_6.2-0
[17] SparseM_1.81      Hmisc_4.6-0       Formula_1.2-4     survival_3.2-13
[21] lattice_0.20-45   forcats_0.5.1     stringr_1.4.0     dplyr_1.0.8
[25] purrr_0.3.4       readr_2.0.2       tidyr_1.2.0       tibble_3.1.6
[29] ggplot2_3.3.5     tidyverse_1.3.1

loaded via a namespace (and not attached):
  [1] readxl_1.3.1        backports_1.3.0      systemfonts_1.0.3
```

```
  [4] plyr_1.8.6           splines_4.1.0        listenv_0.8.0
  [7] TH.data_1.1-0        digest_0.6.29        htmltools_0.5.2
 [10] magick_2.7.3         fansi_1.0.2          magrittr_2.0.2
 [13] checkmate_2.0.0      cluster_2.1.2        tzdb_0.2.0
 [16] recipes_0.1.17       globals_0.14.0       modelr_0.1.8
 [19] gower_0.2.2          matrixStats_0.61.0   extrafontdb_1.0
 [22] sandwich_3.0-1       svglite_2.0.0        jpeg_0.1-9
 [25] colorspace_2.0-3     rvest_1.0.2          mitools_2.4
 [28] haven_2.4.3          xfun_0.30            crayon_1.5.0
 [31] jsonlite_1.8.0       zoo_1.8-9            glue_1.6.2
 [34] gtable_0.3.0         ipred_0.9-12         webshot_0.5.2
 [37] MatrixModels_0.5-0   distributional_0.2.2 car_3.0-12
 [40] Rttf2pt1_1.3.9       future.apply_1.8.1   abind_1.4-5
 [43] scales_1.1.1         mvtnorm_1.1-3        DBI_1.1.1
 [46] rstatix_0.7.0        Rcpp_1.0.8.3         viridisLite_0.4.0
 [49] htmlTable_2.3.0      foreign_0.8-81       stats4_4.1.0
 [52] lava_1.6.10          survey_4.1-1         prodlim_2019.11.13
 [55] htmlwidgets_1.5.4    httr_1.4.2           MatchIt_4.3.0
 [58] RColorBrewer_1.1-2   ellipsis_0.3.2       farver_2.1.0
 [61] pkgconfig_2.0.3      nnet_7.3-16          dbplyr_2.1.1
 [64] utf8_1.2.2           caret_6.0-90         labeling_0.4.2
 [67] tidyselect_1.1.2     rlang_1.0.2          reshape2_1.4.4
 [70] munsell_0.5.0        cellranger_1.1.0     tools_4.1.0
 [73] cli_3.2.0            moments_0.14         generics_0.1.2
 [76] broom_0.7.10         evaluate_0.15        fastmap_1.1.0
 [79] yaml_2.3.5           ModelMetrics_1.2.2.2 knitr_1.37
 [82] fs_1.5.2             nlme_3.1-153         quantreg_5.86
 [85] xml2_1.3.3           compiler_4.1.0       rstudioapi_0.13
 [88] png_0.1-7            ggsignif_0.6.3       reprex_2.0.1
 [91] stringi_1.7.6        nloptr_1.2.2.3       vctrs_0.3.8
 [94] pillar_1.7.0         lifecycle_1.0.1      cowplot_1.1.1
 [97] data.table_1.14.2    conquer_1.2.1        R6_2.5.1
[100] latticeExtra_0.6-29  gridExtra_2.3        parallelly_1.28.1
[103] codetools_0.2-18     polspline_1.1.19     boot_1.3-28
[106] MASS_7.3-54          assertthat_0.2.1     withr_2.5.0
[109] multcomp_1.4-17      hms_1.1.1            grid_4.1.0
[112] rpart_4.1-15         timeDate_3043.102    class_7.3-19
[115] minqa_1.2.4          rmarkdown_2.13       carData_3.0-4
[118] pROC_1.18.0          lubridate_1.8.0      base64enc_0.1-3
```

References

1. Naimi AI, Moodie EEM, Auger N, et al. Constructing inverse probability weights for continuous exposures: a comparison of methods. *Epidemiology (Cambridge, Mass.)*. 2014;25(2):292–299.