

# Supplement 2: 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 2022-11-20 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  $\mu$ ,  $\mu_2$ , we increased the correlation between `mage_g` and  $\mu_2$  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 to nearest 0.1
x1 <- round(15 + mu_un + rnorm(n, 0, sqrt(2)), 1)

# normal exposure distribution, but marginally not normal, but round so it's ordinal to nearest 0.1
x2 <- round(15 + mu_g + rnorm(n, 0, sqrt(2)), 1)

# poisson exposure distribution, but round so it's ordinal to nearest 0.1
x3 <- round(pmax(rpois(n, mu_un) + rnorm(n,0,1), 0), 1)

# normal exposure distribution, but round so it's ordinal to nearest 1
x4 <- round(15 + mu_un + rnorm(n, 0, sqrt(2)))

# normal exposure distribution, but marginally not normal, but round so it's ordinal to nearest 1
x5 <- round(15 + mu_g + rnorm(n, 0, sqrt(2)))

# poisson exposure distribution, but round so it's ordinal to nearest 1
x6 <- round(pmax(rpois(n, mu_un) + rnorm(n,0,1), 0))

# 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) * parity5)))^(-1))

# normal exposure distribution, but marginally not normal
y2 <- rbinom(n, 1, (1 + exp(-(-13 + 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))

# outcome normal exposure distribution, uncorrelated with maternal age
y4 <- rbinom(n, 1, (1 + exp(-(-11.5 + log(1.25) * x4 + 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))

# normal exposure distribution, but marginally not normal
y5 <- rbinom(n, 1, (1 + exp(-(-13 + log(1.25) * x5 + 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
y6 <- rbinom(n, 1, (1 + exp(-(-8.05 + log(1.25) * x6 + 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, x5, x6, n_x1, n_x2, y1, y2, y3, y4, y5, y6, 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 = s

# test exposure distributions
# test <- sim_data(n = 30000)
# hist(test$x1)
# hist(test$x2)
# hist(test$x3)
# hist(test$x4)
# hist(test$x5)
# hist(test$x6)

```

## 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 CPM.

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  $i \in \{2\}$  and binned exposures instead of  $X_i$  when calculating QB weights:

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

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)
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)),
           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)
  x5_formula <- formula(x5 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
  x6_formula <- formula(x6 ~ 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
  x5_ols_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "ps")$weights
  x6_ols_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "ps")$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)$weights
x5_cbgps_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "cbps",
                        over = FALSE)$weights
x6_cbgps_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "cbps",
                        over = FALSE)$weights

#npCBGPS
x1_npcbgps_wts <- weightit(x1_formula, df %>% filter(!is.na(x1)), method = "npcbps",
                        over = FALSE)$weights
x2_npcbgps_wts <- weightit(x2_formula, df %>% filter(!is.na(x2)), method = "npcbps",
                        over = FALSE)$weights
x3_npcbgps_wts <- weightit(x3_formula, df %>% filter(!is.na(x3)), method = "npcbps",
                        over = FALSE)$weights
x4_npcbgps_wts <- weightit(x4_formula, df %>% filter(!is.na(x4)), method = "npcbps",
                        over = FALSE)$weights
x5_npcbgps_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "npcbps",
                        over = FALSE)$weights
x6_npcbgps_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "npcbps",
                        over = FALSE)$weights

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

# only doing QB for x1-x3, because at smaller number of categories, they are the same thing
# 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()

# 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()

# 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()

# 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") %>%

    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()
x5_olr_wts <- orm.wt(object = df %>% filter(!is.na(x5)),
    exposure = "x5",
    cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
    parity5") %>%

    unlist()

```

```

x6_olr_wts <- orm.wt(object = df %>% filter(!is.na(x6)),
  exposure = "x6",
  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, x5_ols_wts, x6_ols_wts,
  x1_cbgps_wts, x2_cbgps_wts, x3_cbgps_wts, x4_cbgps_wts, x5_cbgps_wts, x6_cbgps_wts,
  x1_npcbgps_wts, x2_npcbgps_wts, x3_npcbgps_wts, x4_npcbgps_wts, x5_npcbgps_wts, x6_npcbgps_wts,
  x1_qb10_wts, x2_qb10_wts, x3_qb10_wts,
  x1_qb15_wts, x2_qb15_wts, x3_qb15_wts,
  x1_qb20_wts, x2_qb20_wts, x3_qb20_wts,
  x1_olr_wts, x2_olr_wts, x3_olr_wts, x4_olr_wts, x5_olr_wts, x6_olr_wts)
}

# add in simulation for anything less than 3000, with new seed across all all streams for parallel proc
set.seed(11111, kind = "L'Ecuyer-CMRG")
n_extra <- 3000 - length(sims)
sims2 <- foreach(i = 1:n_extra, .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)),
    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)
  x5_formula <- formula(x5 ~ mage + page + mage*page + parity2 + parity3 + parity4 + parity5)
  x6_formula <- formula(x6 ~ 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
  x5_ols_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "ps")$weights
  x6_ols_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "ps")$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
x5_cbgps_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "cbps",
                        over = FALSE)$weights
x6_cbgps_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "cbps",
                        over = FALSE)$weights

#npCBGPS
x1_npcbgps_wts <- weightit(x1_formula, df %>% filter(!is.na(x1)), method = "npcbps",
                        over = FALSE)$weights
x2_npcbgps_wts <- weightit(x2_formula, df %>% filter(!is.na(x2)), method = "npcbps",
                        over = FALSE)$weights
x3_npcbgps_wts <- weightit(x3_formula, df %>% filter(!is.na(x3)), method = "npcbps",
                        over = FALSE)$weights
x4_npcbgps_wts <- weightit(x4_formula, df %>% filter(!is.na(x4)), method = "npcbps",
                        over = FALSE)$weights
x5_npcbgps_wts <- weightit(x5_formula, df %>% filter(!is.na(x5)), method = "npcbps",
                        over = FALSE)$weights
x6_npcbgps_wts <- weightit(x6_formula, df %>% filter(!is.na(x6)), method = "npcbps",
                        over = FALSE)$weights

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

# only doing QB for x1-x3, because at smaller number of categories, they are the same thing
# 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()

# 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()

# 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()

# 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") %>%

  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()
x5_olr_wts <- orm.wt(object = df %>% filter(!is.na(x5)),
  exposure = "x5",

```

```

        cov_form = "~ mage + page + mage*page + parity2 + parity3 + parity4 +
        parity5") %>%

    unlist()
x6_olr_wts <- orm.wt(object = df %>% filter(!is.na(x6)),
    exposure = "x6",
    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, x5_ols_wts, x6_ols_wts,
    x1_cbgps_wts, x2_cbgps_wts, x3_cbgps_wts, x4_cbgps_wts, x5_cbgps_wts, x6_cbgps_wts,
    x1_npcbgps_wts, x2_npcbgps_wts, x3_npcbgps_wts, x4_npcbgps_wts, x5_npcbgps_wts, x6_npcbgps_wts,
    x1_qb10_wts, x2_qb10_wts, x3_qb10_wts,
    x1_qb15_wts, x2_qb15_wts, x3_qb15_wts,
    x1_qb20_wts, x2_qb20_wts, x3_qb20_wts,
    x1_olr_wts, x2_olr_wts, x3_olr_wts, x4_olr_wts, x5_olr_wts, x6_olr_wts)
}

# combine
sims <- append(sims, sims2)

# save simulation output
Save(sims)

# 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 simulate the truth
set.seed(1111)
df_msm_sim <- sim_data(n = nrow(df))

# 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" | xnum == "x4") {
    alpha <- -11.5
    magevar <- "mage"
  } else if (xnum == "x2" | xnum == "x5") {
    alpha <- -13
    magevar <- "mage_g"
  } else if (xnum == "x3" | xnum == "x6") {
    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
  prob <- plogis(lp)
  mean(prob)
}

# run it with parallel processing

# register clusters (use 6 cores because 6 processes)
registerDoParallel(6)

# now generate weights
msm_truths <- foreach(i = 1:6, .inorder = FALSE, .errorhandling = "remove") %dopar% {
  #exposures
  exps <- c("x1", "x2", "x3", "x4", "x5", "x6")

  # 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)

# 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$x3), prob_x3 = unlist(msm_truths[3]))
x4_truth_key <- tibble(x4 = unique(df_msm_sim$x4), prob_x4 = unlist(msm_truths[4]))
x5_truth_key <- tibble(x5 = unique(df_msm_sim$x5), prob_x5 = unlist(msm_truths[5]))
x6_truth_key <- tibble(x6 = unique(df_msm_sim$x6), prob_x6 = unlist(msm_truths[6]))

# now need to link truths with exposure they match
x_truths <- df_msm_sim %>%
  select(x1, x2, x3, x4, x5, x6)

# 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")
x_truths <- left_join(x_truths, x4_truth_key, by = "x4")
x_truths <- left_join(x_truths, x5_truth_key, by = "x5")
x_truths <- left_join(x_truths, x6_truth_key, by = "x6")

# show that there are no non-missing values of prob_x1:prob_x6 after joining
sum(is.na(x_truths$prob_x1), is.na(x_truths$prob_x2), is.na(x_truths$prob_x3),
  is.na(x_truths$prob_x4), is.na(x_truths$prob_x5), is.na(x_truths$prob_x6))

```

```
[1] 0
```

```

sum(!is.na(x_truths$prob_x1), !is.na(x_truths$prob_x2), !is.na(x_truths$prob_x3),
    !is.na(x_truths$prob_x4), !is.na(x_truths$prob_x5), !is.na(x_truths$prob_x6))/6

[1] 4500000

# 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()

msm_x4 <- glm(x_truths$prob_x4 ~ x_truths$x4, family= binomial)

true_x4 <- coef(msm_x4)["x_truths$x4"] %>% unname()

msm_x5 <- glm(x_truths$prob_x5 ~ x_truths$x5, family= binomial)

true_x5 <- coef(msm_x5)["x_truths$x5"] %>% unname()

msm_x6 <- glm(x_truths$prob_x6 ~ x_truths$x6, family= binomial)

true_x6 <- coef(msm_x6)["x_truths$x6"] %>% 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 | xnum == 4) {
    alpha <- -11.5
    magevar <- "mage"
  } else if (xnum == 2 | xnum == 5) {
    alpha <- -13
    magevar <- "mage_g"
  } else if (xnum == 3 | xnum == 6) {
    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))

# x4
true2_x4_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x4, .x), xnum = 4))

# x5
true2_x5_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x5, .x), xnum = 5))

# x6
true2_x6_qs <- map_dbl(seq(0.1, 0.9, 0.1), ~ getMeanProb(dat = df_msm_sim,
                                                         val = quantile(df_msm_sim$x6, .x), xnum = 6))

```

### Supplemental Table 2.1 - 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)
  x5 <- n_distinct(data$x5)
  x6 <- n_distinct(data$x6)

  data.frame(x1, x2, x3, x4, x5, x6)
}

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

```

### Supplemental Table 2.2 - Dose Response Deciles

```

# 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))

```

	x1	x2	x3	x4	x5	x6
	Min. :76.00	Min. : 96.0	Min. :69.00	Min. : 9.00	Min. :11.0	Min. : 8.00
	1st Qu.:83.00	1st Qu.:105.0	1st Qu.:75.00	1st Qu.:10.00	1st Qu.:13.0	1st Qu.:10.00
	Median :85.00	Median :107.0	Median :76.00	Median :11.00	Median :14.0	Median :10.00
	Mean :85.16	Mean :106.8	Mean :76.54	Mean :10.82	Mean :13.8	Mean :10.04
	3rd Qu.:87.00	3rd Qu.:109.0	3rd Qu.:78.00	3rd Qu.:11.00	3rd Qu.:14.0	3rd Qu.:10.00
	Max. :94.00	Max. :117.0	Max. :85.00	Max. :13.00	Max. :17.0	Max. :12.00

Decile	X1	X2	X3	X4	X5	X6
1	15.1	21.4	0.0	15	21	0
2	15.7	22.3	0.5	16	22	0
3	16.2	22.9	0.9	16	23	1
4	16.6	23.4	1.4	17	23	1
5	17.0	23.9	1.8	17	24	2
6	17.3	24.3	2.3	17	24	2
7	17.7	24.7	2.8	18	25	3
8	18.2	25.3	3.4	18	25	3
9	18.8	25.9	4.3	19	26	4

```

x3_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x3, .x))
x4_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x4, .x))
x5_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x5, .x))
x6_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x6, .x))

# print table of decile values
dec_tab <- tibble(Decile = c(1:9),
                  X1 = x1_quants,
                  X2 = x2_quants,
                  X3 = x3_quants,
                  X4 = x4_quants,
                  X5 = x5_quants,
                  X6 = x6_quants)
kable(dec_tab) %>%
  kable_classic(html_font = "Arial", full_width = FALSE)

```

## Recreating Distributions from Naimi et al.

### Supplemental Table 2.3 - Simulation Descriptive Statistics

```

# 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) - rounded to 0.1",
"X2 (normal, skewed) - rounded to 0.1",
"X3 (Poisson, naimi) - rounded to 0.1",
"X4 (normal, naimi) - rounded to 1",
"X5 (normal, skewed) - rounded to 1",
"X6 (Poisson, naimi) - rounded to 1",
"Y1 (Bernoulli, naimi)",
"Y2 (Bernoulli, skewed)",
"Y3 (Bernoulli, naimi)",
"Y4 (Bernoulli, naimi)",
"Y5 (Bernoulli, skewed)",
"Y6 (Bernoulli, naimi)",
"Naimi Homoscedastic X",
"Naimi Heteroscedastic X",
"Naimi Homoscedastic Y",
"Naimi Heteroscedastic Y"),
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$x4, na.rm = TRUE),
          mean(df$x5, na.rm = TRUE),
          mean(df$x6, na.rm = TRUE),
          mean(df$y1),
          mean(df$y2),
          mean(df$y3, na.rm = TRUE),
          mean(df$y4, na.rm = TRUE),
          mean(df$y5, na.rm = TRUE),
          mean(df$y6, na.rm = TRUE),
          mean(df$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),

```



Variable (Distribution)	Mean	Variance
Maternal Age (normal)	29.84	21.61
Maternal Age (skewed)	30.07	15.46
Paternal Age (normal)	32.52	30.43
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) - rounded to 0.1	16.96	2.16
X2 (normal, skewed) - rounded to 0.1	23.74	3.38
X3 (Poisson, naimi) - rounded to 0.1	2.04	2.67
X4 (normal, naimi) - rounded to 1	16.96	2.24
X5 (normal, skewed) - rounded to 1	23.74	3.45
X6 (Poisson, naimi) - rounded to 1	2.04	2.77
Y1 (Bernoulli, naimi)	0.08	0.07
Y2 (Bernoulli, skewed)	0.08	0.07
Y3 (Bernoulli, naimi)	0.09	0.08
Y4 (Bernoulli, naimi)	0.08	0.07
Y5 (Bernoulli, skewed)	0.08	0.07
Y6 (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

```

var(df$x4, na.rm = TRUE),
var(df$x5, na.rm = TRUE),
var(df$x6, na.rm = TRUE),
var(df$y1),
var(df$y2),
var(df$y3, na.rm = TRUE),
var(df$y4, na.rm = TRUE),
var(df$y5, na.rm = TRUE),
var(df$y6, na.rm = TRUE),
var(df$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 2.1 - Continuous Exposure

```

# now create plot
naimix1 <- ggplot(df, aes(x = n_x1)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, alpha = 0.5, color = "grey50") +

```

```

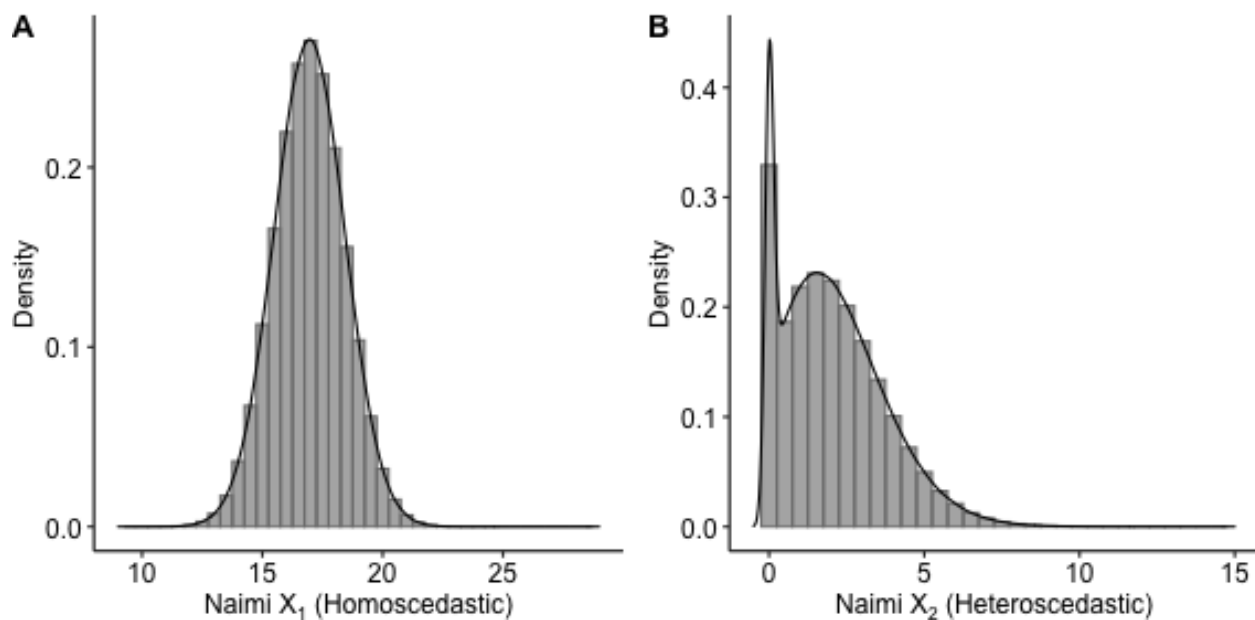
geom_density(adjust = 2) +
ylab("Density") +
scale_x_continuous(name = expression(paste("Naimi ", X[1], " (Homoscedastic)")),
                    limits = c(9, 29), breaks = c(10, 15, 20, 25)) +
theme

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"))

#ggsave("./sim_png/supfig1.1.png")

```



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

## Supplemental Figure 2.2 - 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)

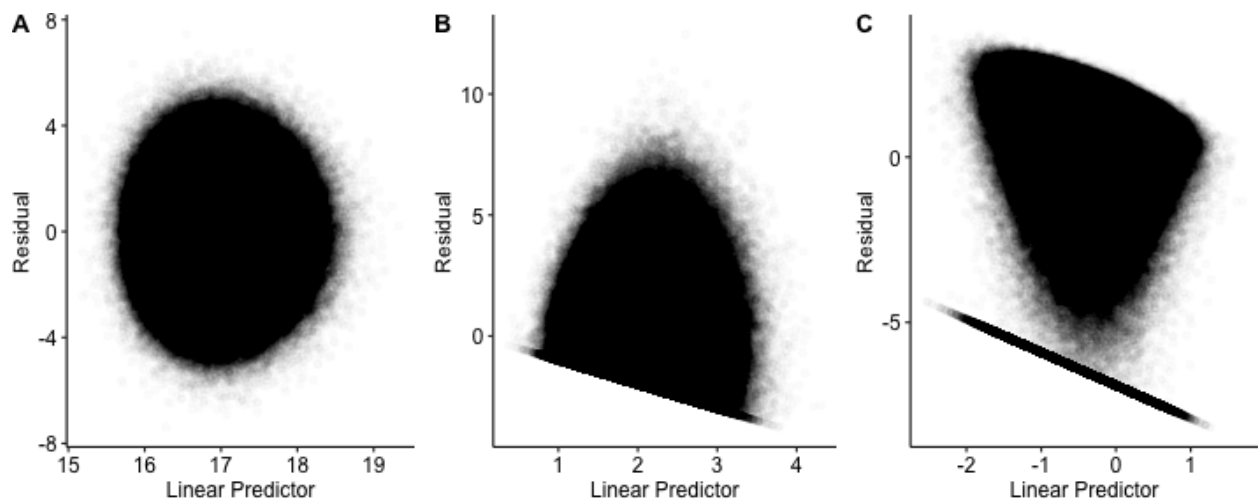
# now plot
x1_plot_n <- ggplot() +
  geom_point(aes(x = preds_x1_n, y = res_x1_n), alpha = 0.01) +
  ylab("Residual") +
  xlab("Linear Predictor") +
  theme

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"))
#ggsave("./sim_png/supfig1.2.png")

```



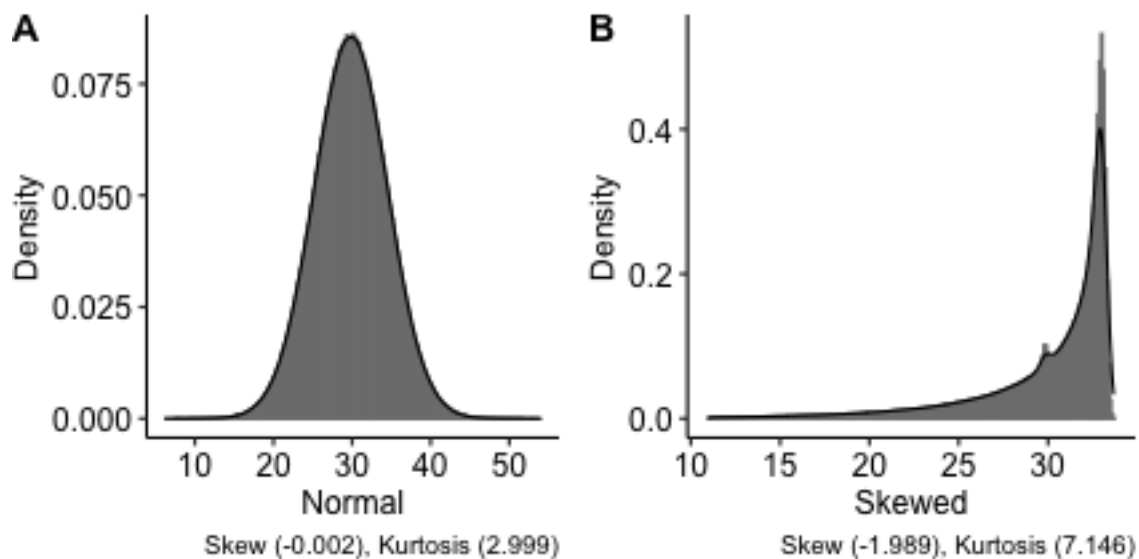
# Updated Distributions

## Supplemental Figure 2.3 - 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"))
#ggsave("./sim_png/supfig1.3.png")
```

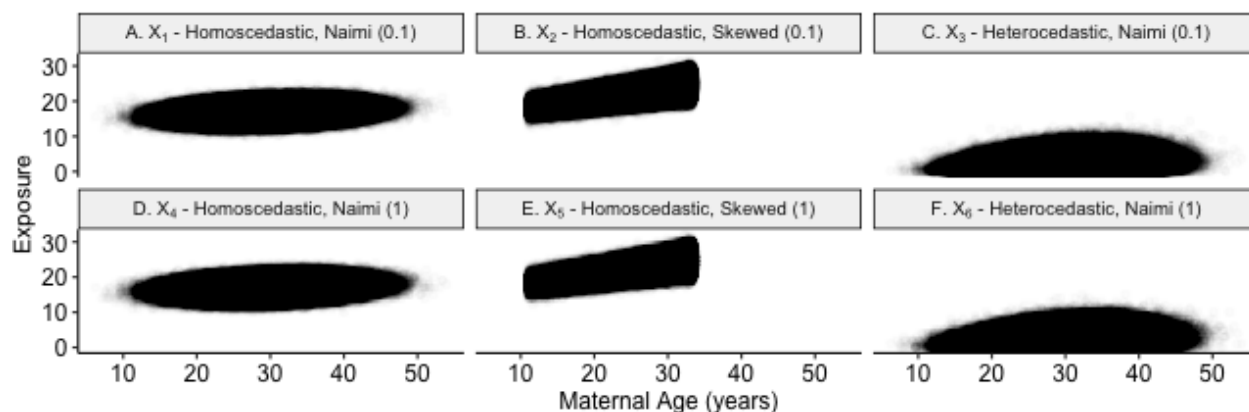


Supplemental Figure 3 shows the updated marginal distributions of maternal age (`mage`) and skewed maternal age (`mage_g`). Despite similar means and variances in `mage` and `mage_g`, `mage` is normally distributed whereas `mage_g` is significantly left-skewed.

## Supplemental Figure 2.4 - Exposure-Covariate Correlations

```
df %>%
  select(mage, mage_g, x1:x6) %>%
  pivot_longer(x1:x6) %>%
  mutate(mage_fin = ifelse(name %in% c("x1", "x3", "x4", "x6"), mage, mage_g)) %>%
  mutate(name = factor(name,
    labels = c(expression(paste("A. ", X[1], " - Homoscedastic, Naimi (0.1)")),
      expression(paste("B. ", X[2], " - Homoscedastic, Skewed (0.1)")),
      expression(paste("C. ", X[3], " - Heterocedastic, Naimi (0.1)")),
      expression(paste("D. ", X[4], " - Homoscedastic, Naimi (1)")),
      expression(paste("E. ", X[5], " - Homoscedastic, Skewed (1)")),
      expression(paste("F. ", X[6], " - Heterocedastic, Naimi (1)"))))) %>%

  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
#ggsave("./sim_png/supfig1.4.png")
```



## Marginal and Conditional Exposure Distributions

Figure 1 - Marginal and Conditional Exposure Distribution

```
# Marginal Exposure Distribution

# now create plot
dens_x1 <- 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]))) +
  # 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))
```

```

dens_x2 <- 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))

dens_x3 <- 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))

dens_x4 <- ggplot(df, aes(x = x4)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, alpha = 0.5, color = "grey50") +
  #geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[4]))) +
  # labs(caption = paste0("Skew (", round(moments::skewness(df$x4), 3),
  #      ")", Kurtosis (", round(moments::kurtosis(df$x4), 3), ")")) +
  theme +
  theme(text = element_text(size = 8.5))

dens_x5 <- ggplot(df, aes(x = x5)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, alpha = 0.5, color = "grey50") +
  #geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[5]))) +
  # labs(caption = paste0("Skew (", round(moments::skewness(df$x5), 3),
  #      ")", Kurtosis (", round(moments::kurtosis(df$x5), 3), ")")) +
  theme +
  theme(text = element_text(size = 8.5))

dens_x6 <- ggplot(df, aes(x = x6)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, alpha = 0.5, color = "grey50") +
  #geom_density(adjust = 2) +
  ylab("Density") +
  scale_x_continuous(name = expression(paste(X[6]))) +
  # labs(caption = " ") +
  theme +
  theme(text = element_text(size = 8.5))

# 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)
ols_x5 <- ols(x5 ~ + mage_g + page + mage_g*page + parity2 + parity3 + parity4 +
  parity5, data = df)
ols_x6 <- ols(x6 ~ + 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)
preds_x5 <- predict(ols_x5)
preds_x6 <- predict(ols_x6)

# residuals
res_x1 <- residuals(ols_x1)
res_x2 <- residuals(ols_x2)
res_x3 <- residuals(ols_x3)
res_x4 <- residuals(ols_x4)
res_x5 <- residuals(ols_x5)
res_x6 <- residuals(ols_x6)

# now plot
x1_plot <- ggplot() +
  geom_point(aes(x = preds_x1, y = res_x1), alpha = 0.01) +
  ylab("Residual") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[1], ")"))) +
  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") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[2], ")"))) +
  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") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[3], ")"))) +
  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") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[4], ")"))) +

```



```

theme +
theme(text = element_text(size = 8))

x5_plot <- ggplot() +
  geom_point(aes(x = preds_x5, y = res_x5), alpha = 0.01) +
  ylab("Residual") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[5], ")"))) +
  theme +
  theme(text = element_text(size = 8))

x6_plot <- ggplot() +
  geom_point(aes(x = preds_x6, y = res_x6), alpha = 0.01) +
  ylab("Residual") +
  scale_x_continuous(name = expression(paste("Linear Predictor (", X[6], ")"))) +
  theme +
  theme(text = element_text(size = 8))

# combine
ggarrange(dens_x1, x1_plot,
          dens_x2, x2_plot,
          dens_x3, x3_plot,
          dens_x4, x4_plot,
          dens_x5, x5_plot,
          dens_x6, x6_plot,
          labels = c("A)", "B)", "C)", "D)", "E)", "F)",
                    "G)", "H)", "I)", "J)", "K)", "L)"),
          nrow = 6, ncol = 2,
          font.label = list(size = 10))

# save plot
ggsave("figs/fig1.tiff", width = 7, height = 7)

```

## Stabilized Inverse Probability Weight Assessments

```

# 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:x6_olr_wts) %>%
  group_by(i) %>%
  summarise(x1_ols_wts = mean(x1_ols_wts),
            x1_cbgps_wts = mean(x1_cbgps_wts),
            x1_npcbgps_wts = mean(x1_npcbgps_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_npcbgps_wts = mean(x2_npcbgps_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_npcbgps_wts = mean(x3_npcbgps_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),
x4_ols_wts = mean(x4_ols_wts),
x4_cbgps_wts = mean(x4_cbgps_wts),
x4_npcbgps_wts = mean(x4_npcbgps_wts),
x4_olr_wts = mean(x4_olr_wts),
x5_ols_wts = mean(x5_ols_wts),
x5_cbgps_wts = mean(x5_cbgps_wts),
x5_npcbgps_wts = mean(x5_npcbgps_wts),
x5_olr_wts = mean(x5_olr_wts),
x6_ols_wts = mean(x6_ols_wts),
x6_cbgps_wts = mean(x6_cbgps_wts),
x6_npcbgps_wts = mean(x6_npcbgps_wts),
x6_olr_wts = mean(x6_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",
                          "Non-parametric covariate balancing generalized propensity score",
                          "Quantile binning categories",
                          "10",
                          "15",
                          "20",
                          "Ordinal logistic regression"),
               `Mean (min, max)` = c(paste0(round(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), ")"),
                                     paste0(round(mean(mean_wts$x1_npcbgps_wts), 2), " (",
                                         round(min(mean_wts$x1_npcbgps_wts), 2), ", ",
                                         round(max(mean_wts$x1_npcbgps_wts), 2), ")"),
                                     NA,
                                     paste0(round(mean(mean_wts$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$x2_cbgps_wts), 2), ")"),
paste0(round(mean(mean_wts$x2_npcbgps_wts), 2), " (",
round(min(mean_wts$x2_npcbgps_wts), 2), ", ",
round(max(mean_wts$x2_npcbgps_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), ")"),
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), ")"),
paste0(round(mean(mean_wts$x3_npcbgps_wts), 2), " (",
round(min(mean_wts$x3_npcbgps_wts), 2), ", ",
round(max(mean_wts$x3_npcbgps_wts), 2), ")"),
NA,
paste0(round(mean(mean_wts$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$x4_cbgps_wts), 2), ")"),
paste0(round(mean(mean_wts$x4_npcbgps_wts), 2), " (",
round(min(mean_wts$x4_npcbgps_wts), 2), ", ",
round(max(mean_wts$x4_npcbgps_wts), 2), ")"),
NA,

```

```

NA,
NA,
NA,
paste0(round(mean(df$x4_olr_wts), 2), " (",
        round(min(df$x4_olr_wts), 2), ", ",
        round(max(df$x4_olr_wts), 2), ")")),
`Mean (min, max)` = c(paste0(round(mean(df$x5_ols_wts), 2), " (",
        round(min(df$x5_ols_wts), 2), ", ",
        round(max(df$x5_ols_wts), 2), ")"),
        paste0(round(mean(df$x5_cbgps_wts), 2), " (",
        round(min(df$x5_cbgps_wts), 2), ", ",
        round(max(df$x5_cbgps_wts), 2), ")"),
        paste0(round(mean(mean_wts$x5_npcbgps_wts), 2), " (",
        round(min(mean_wts$x5_npcbgps_wts), 2), ", ",
        round(max(mean_wts$x5_npcbgps_wts), 2), ")"),
NA,
NA,
NA,
NA,
paste0(round(mean(df$x5_olr_wts), 2), " (",
        round(min(df$x5_olr_wts), 2), ", ",
        round(max(df$x5_olr_wts), 2), ")")),
`Mean (min, max)` = c(paste0(round(mean(df$x6_ols_wts), 2), " (",
        round(min(df$x6_ols_wts), 2), ", ",
        round(max(df$x6_ols_wts), 2), ")"),
        paste0(round(mean(df$x6_cbgps_wts), 2), " (",
        round(min(df$x6_cbgps_wts), 2), ", ",
        round(max(df$x6_cbgps_wts), 2), ")"),
        paste0(round(mean(mean_wts$x6_npcbgps_wts), 2), " (",
        round(min(mean_wts$x6_npcbgps_wts), 2), ", ",
        round(max(mean_wts$x6_npcbgps_wts), 2), ")"),
NA,
NA,
NA,
NA,
paste0(round(mean(df$x6_olr_wts), 2), " (",
        round(min(df$x6_olr_wts), 2), ", ",
        round(max(df$x6_olr_wts), 2), ")"))
)

```

## 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)
x5_formula <- formula(x6 ~ mage_g + page + mage_g*page + parity2 + parity3 + parity4 + parity5)
x6_formula <- formula(x6 ~ 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",
    NPCGPS = "x1_npcbgps_wts",
    CPM = "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",
    NPCGPS = "x2_npcbgps_wts",
    CPM = "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",
    NPCGPS = "x3_npcbgps_wts",
    CPM = "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",
    NPCGPS = "x4_npcbgps_wts",
    CPM = "x4_olr_wts"),
  stats = c("c"),
  un = TRUE, thresholds = c(cor = .1))
bal_tab_x5 <- bal.tab(x5_formula, data = data,
  weights = list(OLS = "x5_ols_wts",
    CBGPS = "x5_cbgps_wts",
    NPCGPS = "x5_npcbgps_wts",
    CPM = "x5_olr_wts"),
  stats = c("c"),
  un = TRUE, thresholds = c(cor = .1))
bal_tab_x6 <- bal.tab(x6_formula, data = data,
  weights = list(OLS = "x6_ols_wts",
    CBGPS = "x6_cbgps_wts",
    NPCGPS = "x6_npcbgps_wts",
    CPM = "x6_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)

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))

# 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)

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))

# 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)

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))

# 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_npcbgps = bal_tab_x1$Balanced.correlations[2, 3],
            x1_qb10 = bal_tab_x1_qb10$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$Balanced.correlations[2, 4],
            x2_uw = sum(bal_tab_x2$Balance$Corr.Un < 0.1),
            x2_ols = bal_tab_x2$Balanced.correlations[2, 1],
            x2_cbgps = bal_tab_x2$Balanced.correlations[2, 2],
            x2_npcbgps = bal_tab_x2$Balanced.correlations[2, 3],
            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, 4],
            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_npcbgps = bal_tab_x3$Balanced.correlations[2, 3],
x3_qb10 = bal_tab_x3_qb10$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$Balanced.correlations[2, 4],
x4_uw = sum(bal_tab_x4$Balance$Corr.Un < 0.1),
x4_ols = bal_tab_x4$Balanced.correlations[2, 1],
x4_cbgps = bal_tab_x4$Balanced.correlations[2, 2],
x4_npcbgps = bal_tab_x4$Balanced.correlations[2, 3],
x4_qb10 = NA,
x4_qb15 = NA,
x4_qb20 = NA,
x4_olr = bal_tab_x4$Balanced.correlations[2, 4],
x5_uw = sum(bal_tab_x5$Balance$Corr.Un < 0.1),
x5_ols = bal_tab_x5$Balanced.correlations[2, 1],
x5_cbgps = bal_tab_x5$Balanced.correlations[2, 2],
x5_npcbgps = bal_tab_x5$Balanced.correlations[2, 3],
x5_qb10 = NA,
x5_qb15 = NA,
x5_qb20 = NA,
x5_olr = bal_tab_x5$Balanced.correlations[2, 4],
x6_uw = sum(bal_tab_x6$Balance$Corr.Un < 0.1),
x6_ols = bal_tab_x6$Balanced.correlations[2, 1],
x6_cbgps = bal_tab_x6$Balanced.correlations[2, 2],
x6_npcbgps = bal_tab_x6$Balanced.correlations[2, 3],
x6_qb10 = NA,
x6_qb15 = NA,
x6_qb20 = NA,
x6_olr = bal_tab_x6$Balanced.correlations[2, 4])
}

# # 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",
                                "Non-parametric 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$x1_cbgps), ")"),
paste0(round(mean(covbal$x1_npcbgps), 2), " (",
                             min(covbal$x1_npcbgps), ", ",
                             max(covbal$x1_npcbgps), ")"),
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$x2_cbgps), ")"),
paste0(round(mean(covbal$x2_npcbgps), 2), " (",
                             min(covbal$x2_npcbgps), ", ",
                             max(covbal$x2_npcbgps), ")"),
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), ")"),
paste0(round(mean(covbal$x3_npcbgps), 2), " (",
        min(covbal$x3_npcbgps), ", ",
        max(covbal$x3_npcbgps), ")"),
NA,
paste0(round(mean(covbal$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_uw), 2), " (",
        min(covbal$x4_uw), ", ",
        max(covbal$x4_uw), ")"),
        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), ")"),
        paste0(round(mean(covbal$x4_npcbgps), 2), " (",
        min(covbal$x4_npcbgps), ", ",
        max(covbal$x4_npcbgps), ")"),
NA,
NA,
NA,
NA,
paste0(round(mean(covbal$x4_olr), 2), " (",
        min(covbal$x4_olr), ", ",
        max(covbal$x4_olr), ")"),
`Mean (min, max)` = c(paste0(round(mean(covbal$x5_uw), 2), " (",
        min(covbal$x5_uw), ", ",
        max(covbal$x5_uw), ")"),
        paste0(round(mean(covbal$x5_ols), 2), " (",
        min(covbal$x5_ols), ", ",
        max(covbal$x5_ols), ")"),
        paste0(round(mean(covbal$x5_cbgps), 2), " (",
        min(covbal$x5_cbgps), ", ",
        max(covbal$x5_cbgps), ")"),
        paste0(round(mean(covbal$x5_npcbgps), 2), " (",

```

```

min(covbal$x5_npcbgps), ", ",
max(covbal$x5_npcbgps), ")"),
NA,
NA,
NA,
NA,
paste0(round(mean(covbal$x5_olr), 2), " (",
min(covbal$x5_olr), ", ",
max(covbal$x5_olr), ")"),
`Mean (min, max)` = c(paste0(round(mean(covbal$x6_uw), 2), " (",
min(covbal$x6_uw), ", ",
max(covbal$x6_uw), ")"),
paste0(round(mean(covbal$x6_ols), 2), " (",
min(covbal$x6_ols), ", ",
max(covbal$x6_ols), ")"),
paste0(round(mean(covbal$x6_cbgps), 2), " (",
min(covbal$x6_cbgps), ", ",
max(covbal$x6_cbgps), ")"),
paste0(round(mean(covbal$x6_npcbgps), 2), " (",
min(covbal$x6_npcbgps), ", ",
max(covbal$x6_npcbgps), ")"),
NA,
NA,
NA,
NA,
paste0(round(mean(covbal$x6_olr), 2), " (",
min(covbal$x6_olr), ", ",
max(covbal$x6_olr), ")"),
)

```

**Table 2 - 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",
  "Non-parametric 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, tab2[3, 2], covbal_tab[4, 2],
  NA,
  NA, tab2[5, 2], covbal_tab[6, 2],
  NA, tab2[6, 2], covbal_tab[7, 2],
  NA, tab2[7, 2], covbal_tab[8, 2],
  NA, tab2[8, 2], covbal_tab[9, 2]),
`Mean (min, max)` = c(NA, NA, covbal_tab[1, 5],
  NA, tab2[1, 5], covbal_tab[2, 5],
  NA, tab2[2, 5], covbal_tab[3, 5],
  NA, tab2[3, 5], covbal_tab[4, 5],
  NA,
  NA, tab2[5, 5], covbal_tab[6, 5],
  NA, tab2[6, 5], covbal_tab[7, 5],
  NA, tab2[7, 5], covbal_tab[8, 5],
  NA, tab2[8, 5], covbal_tab[9, 5]),
`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, tab2[3, 3], covbal_tab[4, 3],
  NA,
  NA, tab2[5, 3], covbal_tab[6, 3],
  NA, tab2[6, 3], covbal_tab[7, 3],
  NA, tab2[7, 3], covbal_tab[8, 3],
  NA, tab2[8, 3], covbal_tab[9, 3]),
`Mean (min, max)` = c(NA, NA, covbal_tab[1, 6],
  NA, tab2[1, 6], covbal_tab[2, 6],
  NA, tab2[2, 6], covbal_tab[3, 6],
  NA, tab2[3, 6], covbal_tab[4, 6],
  NA,
  NA, tab2[5, 6], covbal_tab[6, 6],
  NA, tab2[6, 6], covbal_tab[7, 6],
  NA, tab2[7, 6], covbal_tab[8, 6],
  NA, tab2[8, 6], covbal_tab[9, 6]),
`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, tab2[3, 4], covbal_tab[4, 4],
  NA,
  NA, tab2[5, 4], covbal_tab[6, 4],
  NA, tab2[6, 4], covbal_tab[7, 4],
  NA, tab2[7, 4], covbal_tab[8, 4],
  NA, tab2[8, 4], covbal_tab[9, 4]),
`Mean (min, max)` = c(NA, NA, covbal_tab[1, 7],
  NA, tab2[1, 7], covbal_tab[2, 7],

```

```

NA, tab2[2, 7], covbal_tab[3, 7],
NA, tab2[3, 7], covbal_tab[4, 7],
NA,
NA, tab2[5, 7], covbal_tab[6, 7],
NA, tab2[6, 7], covbal_tab[7, 7],
NA, tab2[7, 7], covbal_tab[8, 7],
NA, tab2[8, 7], covbal_tab[9, 7]))

kable(fin_tab1) %>%
  kable_classic(html_font = "Arial", full_width = FALSE) %>%
  add_header_above(c("Exposure", "X1" = 1, "X4" = 1, "X2" = 1, "X5" = 1,
    "X3" = 1, "X6" = 1), bold = TRUE) %>%
  add_header_above(c("Marginally", "Normal" = 2, "Non-Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 2, "Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_indent(c(2:3, 5:6, 8:9, 11:12, 14:22, 24:25)) %>%
  add_indent(c(2:3, 5:6, 8:9, 11:12, 15:16, 18:19, 21:22, 24:25))

```

## 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
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))

x4_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x4, .x))

x5_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x5, .x))

x6_quants <- map_dbl(seq(0.1, 0.9, 0.1), ~ quantile(df$x6, .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(y4 ~ x4, data = data)
x5_uw <- lrm(y5 ~ x5, data = data)
x6_uw <- lrm(y6 ~ x6, data = data)

# ols
d_ols_x1 <- svydesign(~1, weights = data$x1_ols_wts, data = data)
x1_ols <- svyglm(y1 ~ x1, design = d_ols_x1, family = binomial)
d_ols_x2 <- svydesign(~1, weights = data$x2_ols_wts, data = data)
x2_ols <- svyglm(y2 ~ x2, design = d_ols_x2, family = binomial)
d_ols_x3 <- svydesign(~1, weights = data$x3_ols_wts, data = data)
x3_ols <- svyglm(y3 ~ x3, design = d_ols_x3, family = binomial)
d_ols_x4 <- svydesign(~1, weights = data$x4_ols_wts, data = data)
x4_ols <- svyglm(y4 ~ x4, design = d_ols_x4, family = binomial)
d_ols_x5 <- svydesign(~1, weights = data$x5_ols_wts, data = data)
x5_ols <- svyglm(y5 ~ x5, design = d_ols_x5, family = binomial)
d_ols_x6 <- svydesign(~1, weights = data$x6_ols_wts, data = data)
x6_ols <- svyglm(y6 ~ x6, design = d_ols_x6, family = binomial)

# cbgps
d_cbgps_x1 <- svydesign(~1, weights = data$x1_cbgps_wts, data = data)
x1_cbgps <- svyglm(y1 ~ x1, design = d_cbgps_x1, family = binomial)
d_cbgps_x2 <- svydesign(~1, weights = data$x2_cbgps_wts, data = data)
x2_cbgps <- svyglm(y2 ~ x2, design = d_cbgps_x2, family = binomial)
d_cbgps_x3 <- svydesign(~1, weights = data$x3_cbgps_wts, data = data)
x3_cbgps <- svyglm(y3 ~ x3, design = d_cbgps_x3, family = binomial)
d_cbgps_x4 <- svydesign(~1, weights = data$x4_cbgps_wts, data = data)
x4_cbgps <- svyglm(y4 ~ x4, design = d_cbgps_x4, family = binomial)
d_cbgps_x5 <- svydesign(~1, weights = data$x5_cbgps_wts, data = data)
x5_cbgps <- svyglm(y5 ~ x5, design = d_cbgps_x5, family = binomial)
d_cbgps_x6 <- svydesign(~1, weights = data$x6_cbgps_wts, data = data)
x6_cbgps <- svyglm(y6 ~ x6, design = d_cbgps_x6, family = binomial)

# npcbgps
d_npcbgps_x1 <- svydesign(~1, weights = data$x1_npcbgps_wts, data = data)
x1_npcbgps <- svyglm(y1 ~ x1, design = d_npcbgps_x1, family = binomial)
d_npcbgps_x2 <- svydesign(~1, weights = data$x2_npcbgps_wts, data = data)
x2_npcbgps <- svyglm(y2 ~ x2, design = d_npcbgps_x2, family = binomial)
d_npcbgps_x3 <- svydesign(~1, weights = data$x3_npcbgps_wts, data = data)
x3_npcbgps <- svyglm(y3 ~ x3, design = d_npcbgps_x3, family = binomial)
d_npcbgps_x4 <- svydesign(~1, weights = data$x4_npcbgps_wts, data = data)
x4_npcbgps <- svyglm(y4 ~ x4, design = d_npcbgps_x4, family = binomial)
d_npcbgps_x5 <- svydesign(~1, weights = data$x5_npcbgps_wts, data = data)
x5_npcbgps <- svyglm(y5 ~ x5, design = d_npcbgps_x5, family = binomial)
d_npcbgps_x6 <- svydesign(~1, weights = data$x6_npcbgps_wts, data = data)
x6_npcbgps <- svyglm(y6 ~ x6, design = d_npcbgps_x6, family = binomial)

# qb10
d_qb10_x1 <- svydesign(~1, weights = data$x1_qb10_wts, data = data)
x1_qb10 <- svyglm(y1 ~ x1, design = d_qb10_x1, family = binomial)
d_qb10_x2 <- svydesign(~1, weights = data$x2_qb10_wts, data = data)
x2_qb10 <- svyglm(y2 ~ x2, design = d_qb10_x2, family = binomial)
d_qb10_x3 <- svydesign(~1, weights = data$x3_qb10_wts, data = data)

```

```

x3_qb10 <- svyglm(y3 ~ x3, design = d_qb10_x3, family = binomial)

# qb15
d_qb15_x1 <- svydesign(~1, weights = data$x1_qb15_wts, data = data)
x1_qb15 <- svyglm(y1 ~ x1, design = d_qb15_x1, family = binomial)
d_qb15_x2 <- svydesign(~1, weights = data$x2_qb15_wts, data = data)
x2_qb15 <- svyglm(y2 ~ x2, design = d_qb15_x2, family = binomial)
d_qb15_x3 <- svydesign(~1, weights = data$x3_qb15_wts, data = data)
x3_qb15 <- svyglm(y3 ~ x3, design = d_qb15_x3, family = binomial)

# qb20
d_qb20_x1 <- svydesign(~1, weights = data$x1_qb20_wts, data = data)
x1_qb20 <- svyglm(y1 ~ x1, design = d_qb20_x1, family = binomial)
d_qb20_x2 <- svydesign(~1, weights = data$x2_qb20_wts, data = data)
x2_qb20 <- svyglm(y2 ~ x2, design = d_qb20_x2, family = binomial)
d_qb20_x3 <- svydesign(~1, weights = data$x3_qb20_wts, data = data)
x3_qb20 <- svyglm(y3 ~ x3, design = d_qb20_x3, family = binomial)

# olr
d_olr_x1 <- svydesign(~1, weights = data$x1_olr_wts, data = data)
x1_olr <- svyglm(y1 ~ x1, design = d_olr_x1, family = binomial)
d_olr_x2 <- svydesign(~1, weights = data$x2_olr_wts, data = data)
x2_olr <- svyglm(y2 ~ x2, design = d_olr_x2, family = binomial)
d_olr_x3 <- svydesign(~1, weights = data$x3_olr_wts, data = data)
x3_olr <- svyglm(y3 ~ x3, design = d_olr_x3, family = binomial)
d_olr_x4 <- svydesign(~1, weights = data$x4_olr_wts, data = data)
x4_olr <- svyglm(y4 ~ x4, design = d_olr_x4, family = binomial)
d_olr_x5 <- svydesign(~1, weights = data$x5_olr_wts, data = data)
x5_olr <- svyglm(y5 ~ x5, design = d_olr_x5, family = binomial)
d_olr_x6 <- svydesign(~1, weights = data$x6_olr_wts, data = data)
x6_olr <- svyglm(y6 ~ x6, design = d_olr_x6, family = binomial)

# 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$coefficient[2]
x4_uw_bias <- true_x4 - x4_uw$coefficient[2]
x5_uw_bias <- true_x5 - x5_uw$coefficient[2]
x6_uw_bias <- true_x6 - x6_uw$coefficient[2]

# unweighted se
x1_uw_se <- sqrt(x1_uw$var[2,2])
x2_uw_se <- sqrt(x2_uw$var[2,2])
x3_uw_se <- sqrt(x3_uw$var[2,2])
x4_uw_se <- sqrt(x4_uw$var[2,2])
x5_uw_se <- sqrt(x5_uw$var[2,2])
x6_uw_se <- sqrt(x6_uw$var[2,2])

# unweighted coverage
x1_uw_cov <- (true_x1 > (x1_uw$coefficient[2] - (1.96 * x1_uw_se))) &
  (true_x1 < (x1_uw$coefficient[2] + (1.96 * x1_uw_se)))

```



```

x2_uw_cov <- (true_x2 > (x2_uw$coefficient[2] - (1.96 * x2_uw_se))) &
  (true_x2 < (x2_uw$coefficient[2] + (1.96 * x2_uw_se)))
x3_uw_cov <- (true_x3 > (x3_uw$coefficient[2] - (1.96 * x3_uw_se))) &
  (true_x3 < (x3_uw$coefficient[2] + (1.96 * x3_uw_se)))
x4_uw_cov <- (true_x4 > (x4_uw$coefficient[2] - (1.96 * x4_uw_se))) &
  (true_x4 < (x4_uw$coefficient[2] + (1.96 * x4_uw_se)))
x5_uw_cov <- (true_x5 > (x5_uw$coefficient[2] - (1.96 * x5_uw_se))) &
  (true_x5 < (x5_uw$coefficient[2] + (1.96 * x5_uw_se)))
x6_uw_cov <- (true_x6 > (x6_uw$coefficient[2] - (1.96 * x6_uw_se))) &
  (true_x6 < (x6_uw$coefficient[2] + (1.96 * x6_uw_se)))

# 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$coefficient[2]
x4_ols_bias <- true_x4 - x4_ols$coefficient[2]
x5_ols_bias <- true_x5 - x5_ols$coefficient[2]
x6_ols_bias <- true_x6 - x6_ols$coefficient[2]

# ols se
x1_ols_se <- sqrt(x1_ols$cov.unscaled[2,2])
x2_ols_se <- sqrt(x2_ols$cov.unscaled[2,2])
x3_ols_se <- sqrt(x3_ols$cov.unscaled[2,2])
x4_ols_se <- sqrt(x4_ols$cov.unscaled[2,2])
x5_ols_se <- sqrt(x5_ols$cov.unscaled[2,2])
x6_ols_se <- sqrt(x6_ols$cov.unscaled[2,2])

# ols coverage
x1_ols_cov <- (true_x1 > (x1_ols$coefficient[2] - (1.96 * x1_ols_se))) &
  (true_x1 < (x1_ols$coefficient[2] + (1.96 * x1_ols_se)))
x2_ols_cov <- (true_x2 > (x2_ols$coefficient[2] - (1.96 * x2_ols_se))) &
  (true_x2 < (x2_ols$coefficient[2] + (1.96 * x2_ols_se)))
x3_ols_cov <- (true_x3 > (x3_ols$coefficient[2] - (1.96 * x3_ols_se))) &
  (true_x3 < (x3_ols$coefficient[2] + (1.96 * x3_ols_se)))
x4_ols_cov <- (true_x4 > (x4_ols$coefficient[2] - (1.96 * x4_ols_se))) &
  (true_x4 < (x4_ols$coefficient[2] + (1.96 * x4_ols_se)))
x5_ols_cov <- (true_x5 > (x5_ols$coefficient[2] - (1.96 * x5_ols_se))) &
  (true_x5 < (x5_ols$coefficient[2] + (1.96 * x5_ols_se)))
x6_ols_cov <- (true_x6 > (x6_ols$coefficient[2] - (1.96 * x6_ols_se))) &
  (true_x6 < (x6_ols$coefficient[2] + (1.96 * x6_ols_se)))

# 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$coefficient[2]
x4_cbgps_bias <- true_x4 - x4_cbgps$coefficient[2]
x5_cbgps_bias <- true_x5 - x5_cbgps$coefficient[2]
x6_cbgps_bias <- true_x6 - x6_cbgps$coefficient[2]

# cbgps se
x1_cbgps_se <- sqrt(x1_cbgps$cov.unscaled[2,2])
x2_cbgps_se <- sqrt(x2_cbgps$cov.unscaled[2,2])
x3_cbgps_se <- sqrt(x3_cbgps$cov.unscaled[2,2])

```

```

x4_cbgps_se <- sqrt(x4_cbgps$cov.unscaled[2,2])
x5_cbgps_se <- sqrt(x5_cbgps$cov.unscaled[2,2])
x6_cbgps_se <- sqrt(x6_cbgps$cov.unscaled[2,2])

# cbgps coverage
x1_cbgps_cov <- (true_x1 > (x1_cbgps$coefficient[2] - (1.96 * x1_cbgps_se))) &
  (true_x1 < (x1_cbgps$coefficient[2] + (1.96 * x1_cbgps_se)))
x2_cbgps_cov <- (true_x2 > (x2_cbgps$coefficient[2] - (1.96 * x2_cbgps_se))) &
  (true_x2 < (x2_cbgps$coefficient[2] + (1.96 * x2_cbgps_se)))
x3_cbgps_cov <- (true_x3 > (x3_cbgps$coefficient[2] - (1.96 * x3_cbgps_se))) &
  (true_x3 < (x3_cbgps$coefficient[2] + (1.96 * x3_cbgps_se)))
x4_cbgps_cov <- (true_x4 > (x4_cbgps$coefficient[2] - (1.96 * x4_cbgps_se))) &
  (true_x4 < (x4_cbgps$coefficient[2] + (1.96 * x4_cbgps_se)))
x5_cbgps_cov <- (true_x5 > (x5_cbgps$coefficient[2] - (1.96 * x5_cbgps_se))) &
  (true_x5 < (x5_cbgps$coefficient[2] + (1.96 * x5_cbgps_se)))
x6_cbgps_cov <- (true_x6 > (x6_cbgps$coefficient[2] - (1.96 * x6_cbgps_se))) &
  (true_x6 < (x6_cbgps$coefficient[2] + (1.96 * x6_cbgps_se)))

# npcbgps
x1_npcbgps_bias <- true_x1 - x1_npcbgps$coefficient[2]
x2_npcbgps_bias <- true_x2 - x2_npcbgps$coefficient[2]
x3_npcbgps_bias <- true_x3 - x3_npcbgps$coefficient[2]
x4_npcbgps_bias <- true_x4 - x4_npcbgps$coefficient[2]
x5_npcbgps_bias <- true_x5 - x5_npcbgps$coefficient[2]
x6_npcbgps_bias <- true_x6 - x6_npcbgps$coefficient[2]

# npcbgps se
x1_npcbgps_se <- sqrt(x1_npcbgps$cov.unscaled[2,2])
x2_npcbgps_se <- sqrt(x2_npcbgps$cov.unscaled[2,2])
x3_npcbgps_se <- sqrt(x3_npcbgps$cov.unscaled[2,2])
x4_npcbgps_se <- sqrt(x4_npcbgps$cov.unscaled[2,2])
x5_npcbgps_se <- sqrt(x5_npcbgps$cov.unscaled[2,2])
x6_npcbgps_se <- sqrt(x6_npcbgps$cov.unscaled[2,2])

# npcbgps coverage
x1_npcbgps_cov <- (true_x1 > (x1_npcbgps$coefficient[2] - (1.96 * x1_npcbgps_se))) &
  (true_x1 < (x1_npcbgps$coefficient[2] + (1.96 * x1_npcbgps_se)))
x2_npcbgps_cov <- (true_x2 > (x2_npcbgps$coefficient[2] - (1.96 * x2_npcbgps_se))) &
  (true_x2 < (x2_npcbgps$coefficient[2] + (1.96 * x2_npcbgps_se)))
x3_npcbgps_cov <- (true_x3 > (x3_npcbgps$coefficient[2] - (1.96 * x3_npcbgps_se))) &
  (true_x3 < (x3_npcbgps$coefficient[2] + (1.96 * x3_npcbgps_se)))
x4_npcbgps_cov <- (true_x4 > (x4_npcbgps$coefficient[2] - (1.96 * x4_npcbgps_se))) &
  (true_x4 < (x4_npcbgps$coefficient[2] + (1.96 * x4_npcbgps_se)))
x5_npcbgps_cov <- (true_x5 > (x5_npcbgps$coefficient[2] - (1.96 * x5_npcbgps_se))) &
  (true_x5 < (x5_npcbgps$coefficient[2] + (1.96 * x5_npcbgps_se)))
x6_npcbgps_cov <- (true_x6 > (x6_npcbgps$coefficient[2] - (1.96 * x6_npcbgps_se))) &
  (true_x6 < (x6_npcbgps$coefficient[2] + (1.96 * x6_npcbgps_se)))

# 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]

```

```

# qb10 se
x1_qb10_se <- sqrt(x1_qb10$cov.unscaled[2,2])
x2_qb10_se <- sqrt(x2_qb10$cov.unscaled[2,2])
x3_qb10_se <- sqrt(x3_qb10$cov.unscaled[2,2])

# qb10 coverage
x1_qb10_cov <- (true_x1 > (x1_qb10$coefficient[2] - (1.96 * x1_qb10_se))) &
  (true_x1 < (x1_qb10$coefficient[2] + (1.96 * x1_qb10_se)))
x2_qb10_cov <- (true_x2 > (x2_qb10$coefficient[2] - (1.96 * x2_qb10_se))) &
  (true_x2 < (x2_qb10$coefficient[2] + (1.96 * x2_qb10_se)))
x3_qb10_cov <- (true_x3 > (x3_qb10$coefficient[2] - (1.96 * x3_qb10_se))) &
  (true_x3 < (x3_qb10$coefficient[2] + (1.96 * x3_qb10_se)))

# 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]

# qb15 se
x1_qb15_se <- sqrt(x1_qb15$cov.unscaled[2,2])
x2_qb15_se <- sqrt(x2_qb15$cov.unscaled[2,2])
x3_qb15_se <- sqrt(x3_qb15$cov.unscaled[2,2])

# qb15 coverage
x1_qb15_cov <- (true_x1 > (x1_qb15$coefficient[2] - (1.96 * x1_qb15_se))) &
  (true_x1 < (x1_qb15$coefficient[2] + (1.96 * x1_qb15_se)))
x2_qb15_cov <- (true_x2 > (x2_qb15$coefficient[2] - (1.96 * x2_qb15_se))) &
  (true_x2 < (x2_qb15$coefficient[2] + (1.96 * x2_qb15_se)))
x3_qb15_cov <- (true_x3 > (x3_qb15$coefficient[2] - (1.96 * x3_qb15_se))) &
  (true_x3 < (x3_qb15$coefficient[2] + (1.96 * x3_qb15_se)))

# 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]

# qb20 se
x1_qb20_se <- sqrt(x1_qb20$cov.unscaled[2,2])
x2_qb20_se <- sqrt(x2_qb20$cov.unscaled[2,2])
x3_qb20_se <- sqrt(x3_qb20$cov.unscaled[2,2])

# qb20 coverage
x1_qb20_cov <- (true_x1 > (x1_qb20$coefficient[2] - (1.96 * x1_qb20_se))) &
  (true_x1 < (x1_qb20$coefficient[2] + (1.96 * x1_qb20_se)))
x2_qb20_cov <- (true_x2 > (x2_qb20$coefficient[2] - (1.96 * x2_qb20_se))) &
  (true_x2 < (x2_qb20$coefficient[2] + (1.96 * x2_qb20_se)))
x3_qb20_cov <- (true_x3 > (x3_qb20$coefficient[2] - (1.96 * x3_qb20_se))) &
  (true_x3 < (x3_qb20$coefficient[2] + (1.96 * x3_qb20_se)))

# 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_x4 - x4_olr$coefficient[2]
x5_olr_bias <- true_x5 - x5_olr$coefficient[2]
x6_olr_bias <- true_x6 - x6_olr$coefficient[2]

# olr se
x1_olr_se <- sqrt(x1_olr$cov.unscaled[2,2])
x2_olr_se <- sqrt(x2_olr$cov.unscaled[2,2])
x3_olr_se <- sqrt(x3_olr$cov.unscaled[2,2])
x4_olr_se <- sqrt(x4_olr$cov.unscaled[2,2])
x5_olr_se <- sqrt(x5_olr$cov.unscaled[2,2])
x6_olr_se <- sqrt(x6_olr$cov.unscaled[2,2])

# olr coverage
x1_olr_cov <- (true_x1 > (x1_olr$coefficient[2] - (1.96 * x1_olr_se))) &
  (true_x1 < (x1_olr$coefficient[2] + (1.96 * x1_olr_se)))
x2_olr_cov <- (true_x2 > (x2_olr$coefficient[2] - (1.96 * x2_olr_se))) &
  (true_x2 < (x2_olr$coefficient[2] + (1.96 * x2_olr_se)))
x3_olr_cov <- (true_x3 > (x3_olr$coefficient[2] - (1.96 * x3_olr_se))) &
  (true_x3 < (x3_olr$coefficient[2] + (1.96 * x3_olr_se)))
x4_olr_cov <- (true_x4 > (x4_olr$coefficient[2] - (1.96 * x4_olr_se))) &
  (true_x4 < (x4_olr$coefficient[2] + (1.96 * x4_olr_se)))
x5_olr_cov <- (true_x5 > (x5_olr$coefficient[2] - (1.96 * x5_olr_se))) &
  (true_x5 < (x5_olr$coefficient[2] + (1.96 * x5_olr_se)))
x6_olr_cov <- (true_x6 > (x6_olr$coefficient[2] - (1.96 * x6_olr_se))) &
  (true_x6 < (x6_olr$coefficient[2] + (1.96 * x6_olr_se)))

# 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(x4_quants,
  ~ predict(x4_uw,
    newdata = .x,
    type = "fitted"))
x5_uw_qs <- map_dbl(x5_quants,
  ~ predict(x5_uw,
    newdata = .x,
    type = "fitted"))
x6_uw_qs <- map_dbl(x6_quants,
  ~ predict(x6_uw,

```

```

                                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_x4_qs - x4_uw_qs

# x5
x5_uw_bias2 <- true2_x5_qs - x5_uw_qs

# x6
x6_uw_bias2 <- true2_x6_qs - x6_uw_qs

# ols
x1_ols_qs <- data.frame(predict(x1_ols,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_ols_qs <- data.frame(predict(x2_ols,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_ols_qs <- data.frame(predict(x3_ols,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response
x4_ols_qs <- data.frame(predict(x4_ols,
                                newdata = data.frame(x4 = x4_quants),
                                type = "response"))$response
x5_ols_qs <- data.frame(predict(x5_ols,
                                newdata = data.frame(x5 = x5_quants),
                                type = "response"))$response
x6_ols_qs <- data.frame(predict(x6_ols,
                                newdata = data.frame(x6 = x6_quants),
                                type = "response"))$response

# x1
x1_ols_bias2 <- true2_x1_qs - x1_ols_qs

# x2
x2_ols_bias2 <- true2_x2_qs - x2_ols_qs

# x3
x3_ols_bias2 <- true2_x3_qs - x3_ols_qs

# x4
x4_ols_bias2 <- true2_x4_qs - x4_ols_qs

```

```

# x5
x5_ols_bias2 <- true2_x5_qs - x5_ols_qs

# x6
x6_ols_bias2 <- true2_x6_qs - x6_ols_qs

# cbgps
x1_cbgps_qs <- data.frame(predict(x1_cbgps,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_cbgps_qs <- data.frame(predict(x2_cbgps,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_cbgps_qs <- data.frame(predict(x3_cbgps,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response
x4_cbgps_qs <- data.frame(predict(x4_cbgps,
                                newdata = data.frame(x4 = x4_quants),
                                type = "response"))$response
x5_cbgps_qs <- data.frame(predict(x5_cbgps,
                                newdata = data.frame(x5 = x5_quants),
                                type = "response"))$response
x6_cbgps_qs <- data.frame(predict(x6_cbgps,
                                newdata = data.frame(x6 = x6_quants),
                                type = "response"))$response

# 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_x4_qs - x4_cbgps_qs

# x5
x5_cbgps_bias2 <- true2_x5_qs - x5_cbgps_qs

# x6
x6_cbgps_bias2 <- true2_x6_qs - x6_cbgps_qs

# npcbgps
x1_npcbgps_qs <- data.frame(predict(x1_npcbgps,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_npcbgps_qs <- data.frame(predict(x2_npcbgps,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_npcbgps_qs <- data.frame(predict(x3_npcbgps,
                                newdata = data.frame(x3 = x3_quants),

```

```

                                type = "response"))$response
x4_npcbgps_qs <- data.frame(predict(x4_npcbgps,
                                newdata = data.frame(x4 = x4_quants),
                                type = "response"))$response
x5_npcbgps_qs <- data.frame(predict(x5_npcbgps,
                                newdata = data.frame(x5 = x5_quants),
                                type = "response"))$response
x6_npcbgps_qs <- data.frame(predict(x6_npcbgps,
                                newdata = data.frame(x6 = x6_quants),
                                type = "response"))$response

# x1
x1_npcbgps_bias2 <- true2_x1_qs - x1_npcbgps_qs

# x2
x2_npcbgps_bias2 <- true2_x2_qs - x2_npcbgps_qs

# x3
x3_npcbgps_bias2 <- true2_x3_qs - x3_npcbgps_qs

# x4
x4_npcbgps_bias2 <- true2_x4_qs - x4_npcbgps_qs

# x5
x5_npcbgps_bias2 <- true2_x5_qs - x5_npcbgps_qs

# x6
x6_npcbgps_bias2 <- true2_x6_qs - x6_npcbgps_qs

# qb10
x1_qb10_qs <- data.frame(predict(x1_qb10,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_qb10_qs <- data.frame(predict(x2_qb10,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_qb10_qs <- data.frame(predict(x3_qb10,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response

# 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

# qb15
x1_qb15_qs <- data.frame(predict(x1_qb15,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response

```

```

x2_qb15_qs <- data.frame(predict(x2_qb15,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_qb15_qs <- data.frame(predict(x3_qb15,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response

# 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

# qb20
x1_qb20_qs <- data.frame(predict(x1_qb20,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_qb20_qs <- data.frame(predict(x2_qb20,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_qb20_qs <- data.frame(predict(x3_qb20,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response

# 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

# olr
x1_olr_qs <- data.frame(predict(x1_olr,
                                newdata = data.frame(x1 = x1_quants),
                                type = "response"))$response
x2_olr_qs <- data.frame(predict(x2_olr,
                                newdata = data.frame(x2 = x2_quants),
                                type = "response"))$response
x3_olr_qs <- data.frame(predict(x3_olr,
                                newdata = data.frame(x3 = x3_quants),
                                type = "response"))$response
x4_olr_qs <- data.frame(predict(x4_olr,
                                newdata = data.frame(x4 = x4_quants),
                                type = "response"))$response
x5_olr_qs <- data.frame(predict(x5_olr,
                                newdata = data.frame(x5 = x5_quants),
                                type = "response"))$response
x6_olr_qs <- data.frame(predict(x6_olr,

```



```

newdata = data.frame(x6 = x6_quants),
type = "response"))$response

# 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_x4_qs - x4_olr_qs

# x5
x5_olr_bias2 <- true2_x5_qs - x5_olr_qs

# x6
x6_olr_bias2 <- true2_x6_qs - x6_olr_qs

# output dataframe
bias1 <- data.frame(i,
  x1_uw_bias, x2_uw_bias, x3_uw_bias, x4_uw_bias, x5_uw_bias, x6_uw_bias,
  x1_ols_bias, x2_ols_bias, x3_ols_bias, x4_ols_bias, x5_ols_bias, x6_ols_bias,
  x1_cbgps_bias, x2_cbgps_bias, x3_cbgps_bias, x4_cbgps_bias, x5_cbgps_bias, x6_cbgps_bias,
  x1_npcbgps_bias, x2_npcbgps_bias, x3_npcbgps_bias, x4_npcbgps_bias, x5_npcbgps_bias, x6_npcbgps_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, x4_olr_bias, x5_olr_bias, x6_olr_bias)

se1 <- data.frame(i,
  x1_uw_se, x2_uw_se, x3_uw_se, x4_uw_se, x5_uw_se, x6_uw_se,
  x1_ols_se, x2_ols_se, x3_ols_se, x4_ols_se, x5_ols_se, x6_ols_se,
  x1_cbgps_se, x2_cbgps_se, x3_cbgps_se, x4_cbgps_se, x5_cbgps_se, x6_cbgps_se,
  x1_npcbgps_se, x2_npcbgps_se, x3_npcbgps_se, x4_npcbgps_se, x5_npcbgps_se, x6_npcbgps_se,
  x1_qb10_se, x2_qb10_se, x3_qb10_se,
  x1_qb15_se, x2_qb15_se, x3_qb15_se,
  x1_qb20_se, x2_qb20_se, x3_qb20_se,
  x1_olr_se, x2_olr_se, x3_olr_se, x4_olr_se, x5_olr_se, x6_olr_se)

cov1 <- data.frame(i,
  x1_uw_cov, x2_uw_cov, x3_uw_cov, x4_uw_cov, x5_uw_cov, x6_uw_cov,
  x1_ols_cov, x2_ols_cov, x3_ols_cov, x4_ols_cov, x5_ols_cov, x6_ols_cov,
  x1_cbgps_cov, x2_cbgps_cov, x3_cbgps_cov, x4_cbgps_cov, x5_cbgps_cov, x6_cbgps_cov,
  x1_npcbgps_cov, x2_npcbgps_cov, x3_npcbgps_cov, x4_npcbgps_cov, x5_npcbgps_cov, x6_npcbgps_cov,
  x1_qb10_cov, x2_qb10_cov, x3_qb10_cov,
  x1_qb15_cov, x2_qb15_cov, x3_qb15_cov,
  x1_qb20_cov, x2_qb20_cov, x3_qb20_cov,
  x1_olr_cov, x2_olr_cov, x3_olr_cov, x4_olr_cov, x5_olr_cov, x6_olr_cov)

bias2 <- data.frame(quantile = c(1:9),

```

```

x1_uw_bias2, x2_uw_bias2, x3_uw_bias2, x4_uw_bias2, x5_uw_bias2, x6_uw_bias2,
x1_ols_bias2, x2_ols_bias2, x3_ols_bias2, x4_ols_bias2, x5_ols_bias2, x6_ols_bias2,
x1_cbgps_bias2, x2_cbgps_bias2, x3_cbgps_bias2, x4_cbgps_bias2, x5_cbgps_bias2, x6_cbgps_b
x1_npcbgps_bias2, x2_npcbgps_bias2, x3_npcbgps_bias2, x4_npcbgps_bias2, x5_npcbgps_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, x4_olr_bias2, x5_olr_bias2, x6_olr_bias2) %>%
pivot_wider(names_from = quantile,
            values_from = x1_uw_bias2:x6_olr_bias2)

data.frame(bias1, se1, cov1, 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)

```

## Dose Reponse Decile Approach

### Supplemental Figure 2.5 - 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_npcbgps_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_npcbgps_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_npcbgps_bias2_", decile),
      paste0("x3_qb10_bias2_", decile),
      paste0("x3_qb15_bias2_", decile),
      paste0("x3_qb20_bias2_", decile),
      paste0("x3_olr_bias2_", decile))))

x4_bias2 <- bias %>%
  select(starts_with("x4") & contains("bias2") & ends_with(as.character(decile))) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
    levels = c(paste0("x4_uw_bias2_", decile),
      paste0("x4_ols_bias2_", decile),
      paste0("x4_cbgps_bias2_", decile),
      paste0("x4_npcbgps_bias2_", decile),
      paste0("x4_olr_bias2_", decile))))

x5_bias2 <- bias %>%
  select(starts_with("x5") & contains("bias2") & ends_with(as.character(decile))) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
    levels = c(paste0("x5_uw_bias2_", decile),
      paste0("x5_ols_bias2_", decile),
      paste0("x5_cbgps_bias2_", decile),
      paste0("x5_npcbgps_bias2_", decile),
      paste0("x5_olr_bias2_", decile))))

x6_bias2 <- bias %>%
  select(starts_with("x6") & contains("bias2") & ends_with(as.character(decile))) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
    levels = c(paste0("x6_uw_bias2_", decile),
      paste0("x6_ols_bias2_", decile),
      paste0("x6_cbgps_bias2_", decile),
      paste0("x6_npcbgps_bias2_", decile),
      paste0("x6_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("CPM", "QB20", "QB15", "QB10", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),

```

```

        breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
theme +
theme(axis.text = element_text(size = 20),
      axis.title = element_text(size = 24))

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("CPM", "QB20", "QB15", "QB10", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),
                    breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
  theme +
  theme(axis.text = element_text(size = 20),
        axis.title = element_text(size = 24))

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("CPM", "QB20", "QB15", "QB10", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),
                    breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
  theme +
  theme(axis.text = element_text(size = 20),
        axis.title = element_text(size = 24))

x4_bias_plot2 <- ggplot(x4_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("CPM", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),
                    breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
  theme +
  theme(axis.text = element_text(size = 20),
        axis.title = element_text(size = 24))

x5_bias_plot2 <- ggplot(x5_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("CPM", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),
                    breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
  theme +
  theme(axis.text = element_text(size = 20),
        axis.title = element_text(size = 24))

x6_bias_plot2 <- ggplot(x6_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("CPM", "npCBGPS", "CBGPS", "OLS", "UW")) +
  scale_x_continuous(name = "Bias", limits = c(-0.125, 0.125),
                    breaks = seq(-0.1, 0.1, 0.05), labels = seq(-0.1, 0.1, 0.05)) +
  theme +
  theme(axis.text = element_text(size = 20),

```

```

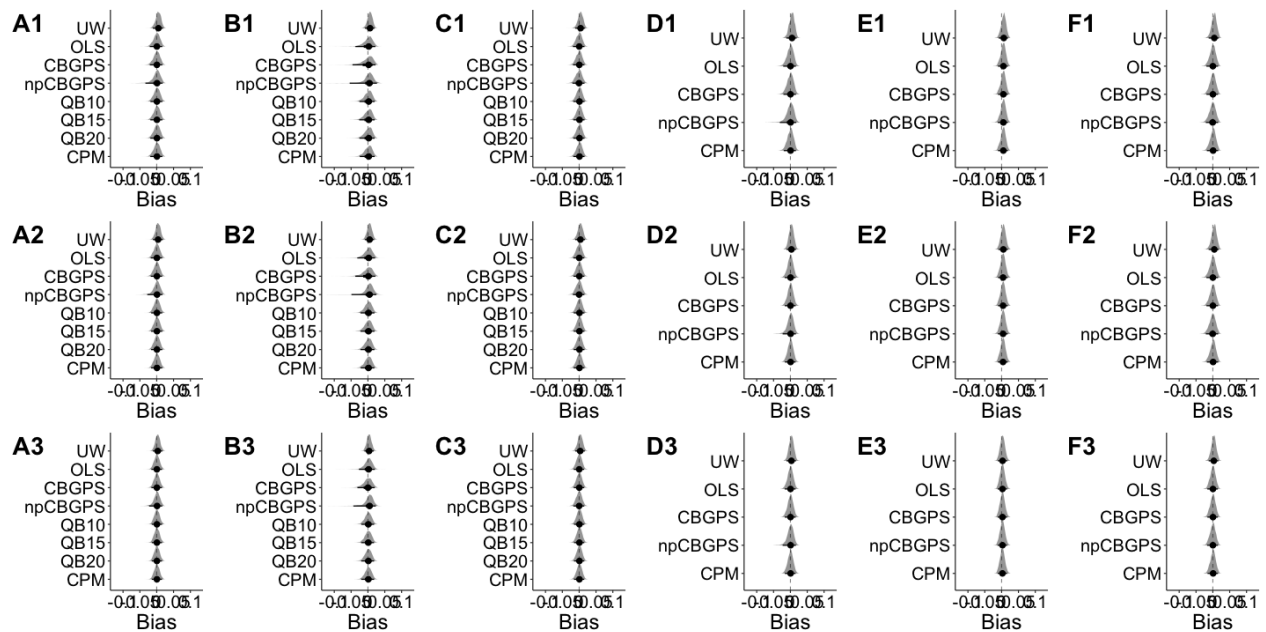
axis.title = element_text(size = 24))

# combine plots
ggarrange(x1_bias_plot2, x2_bias_plot2, x3_bias_plot2,
  x4_bias_plot2, x5_bias_plot2, x6_bias_plot2,
  labels = c(paste0("A", decile), paste0("B", decile), paste0("C", decile),
    paste0("D", decile), paste0("E", decile), paste0("F", decile)),
  font.label = list(size = 28),
  nrow = 1)
}

# plot all deciles
aus_plots <- map(c(1:9), ~ aus_bias_plot(.x))

# plots in chunks
ggarrange(aus_plots[[1]], aus_plots[[2]], aus_plots[[3]], ncol = 1)
#ggsave("./sim_png/supfig1.5_1.png")
ggarrange(aus_plots[[4]], aus_plots[[5]], aus_plots[[6]], ncol = 1)
#ggsave("./sim_png/supfig1.5_2.png")
ggarrange(aus_plots[[7]], aus_plots[[8]], aus_plots[[9]], ncol = 1)
#ggsave("./sim_png/supfig1.5_3.png")

```



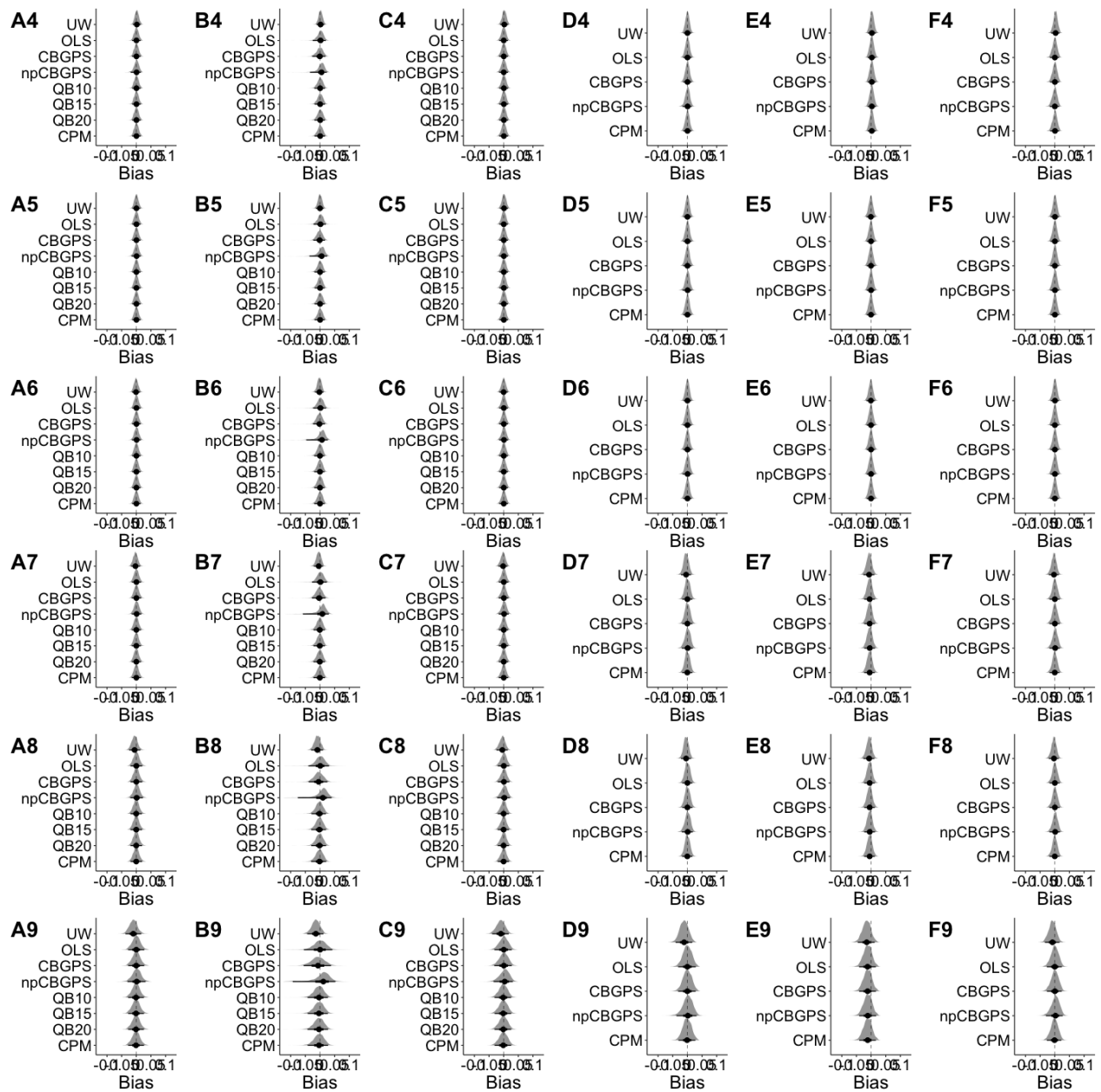


Figure legend: Column A) Homoscedastic Exposure, Conditionally Normal, Marginally Normal, Column B) Homoscedastic Exposure, Conditionally Normal, Marginally Non-Normal, Column C) Heteroscedastic Exposure, Conditionally Non-Normal, Marginally Non-Normal. The number next to each panel represents the decile,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

#### Supplemental Figure 2.6 - Mean Squared Error at Deciles

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

```

        "npCBGPS",
        "QB10",
        "QB15",
        "QB20",
        "OLR"),
    Exposure = c("x1",
                  "x2",
                  "x3",
                  "x4",
                  "x5",
                  "x6"),
    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)
}

# Marginal Log Odds Ratio
mlor_mse <- expand_grid(Decile = c(0),
                       Method = c("UW",
                                   "OLS",
                                   "CBGPS",
                                   "npCBGPS",
                                   "QB10",
                                   "QB15",
                                   "QB20",
                                   "OLR"),
                       Exposure = c("x1",
                                    "x2",
                                    "x3",
                                    "x4",
                                    "x5",
                                    "x6"),
                       MSE = NA)

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

# now combine
mse_plot <- bind_rows(austin_mse, mlor_mse) %>%
  mutate(Method = ifelse(Method == "OLR", "CPM", Method),
         Type = ifelse(Decile == 0, "MLOR", "Decile"))

```

```

# now make plot

# first update levels
mse_plot <- mse_plot %>%
  mutate(Decile = factor(Decile),
         Method = factor(Method, levels = c("UW",
                                             "QB10",
                                             "OLS",
                                             "QB15",
                                             "CBGPS",
                                             "QB20",
                                             "npCBGPS",
                                             "CPM")),
         Exposure = factor(Exposure, levels = c("x1",
                                                  "x2",
                                                  "x3",
                                                  "x4",
                                                  "x5",
                                                  "x6")))) %>%

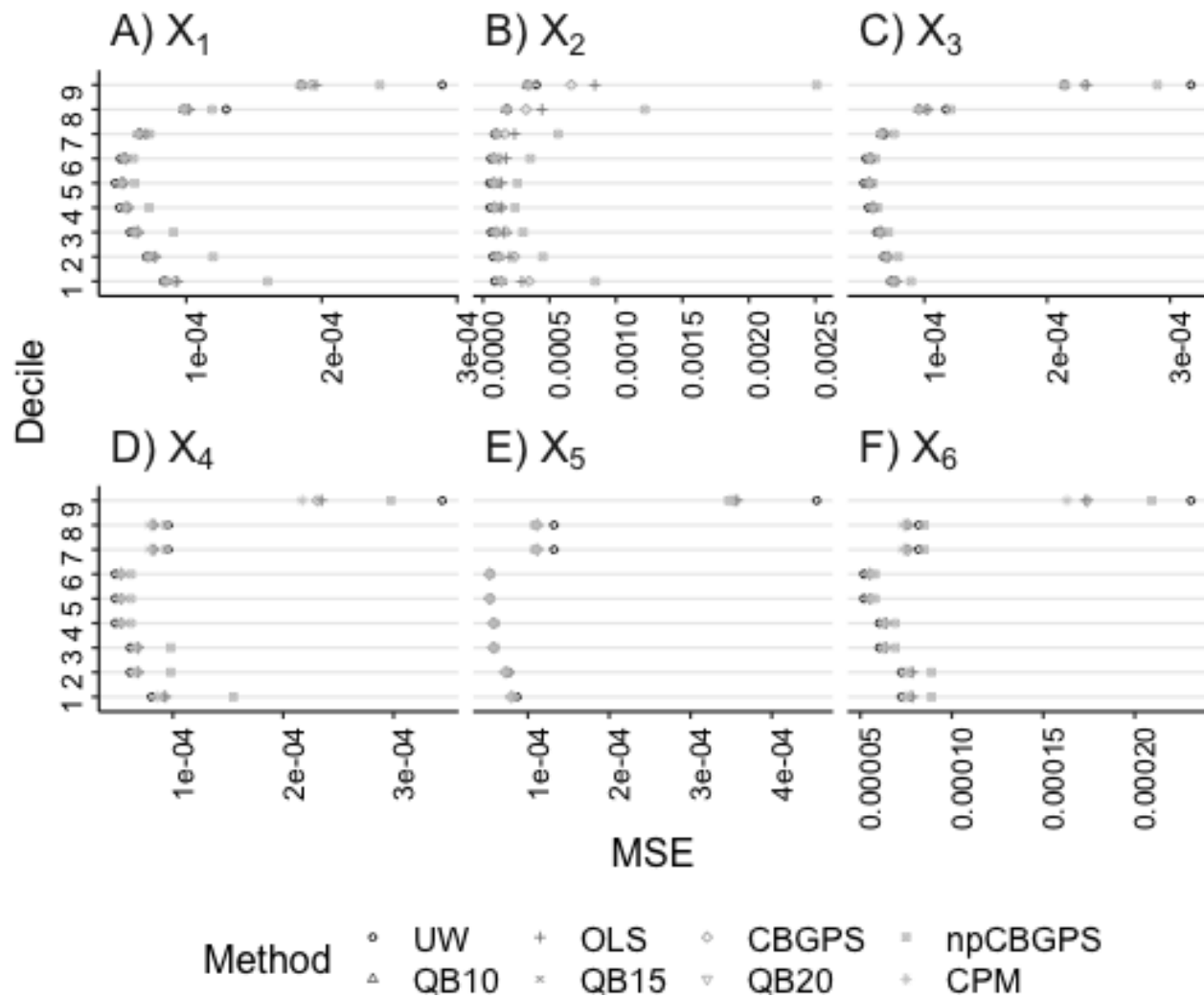
  mutate(name = factor(Exposure,
                      labels = c(expression(paste("A " , X[1])),
                                expression(paste("B " , X[2])),
                                expression(paste("C " , X[3])),
                                expression(paste("D " , X[4])),
                                expression(paste("E " , X[5])),
                                expression(paste("F " , X[6])))))

# fig 2
mse_plot %>% filter(Decile != 0) %>%
  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 = 3, labeller = "label_parsed", scales = "free_x") +
  scale_color_grey(guide = guide_legend(nrow = 2)) +
  scale_shape_manual(values = 1:nlevels(mse_plot$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 = 12, angle = 90),
        axis.title = element_text(size = 16),
        legend.title = element_text(size = 16),
        legend.text = element_text(size = 14),
        strip.text = element_text(hjust = 0, size = 18),
        strip.background = element_blank())

# save plot
#ggsave("./sim_png/supfig1.6.png")

```





## Marginal Log Odds Ratio Approach

Figure 2 - Marginal Log Odds Ratio Approach Bias

```
# make df of x1:4 biases
x1_bias <- bias %>%
  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_npcbgps_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_npcbgps_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_npcbgps_bias",
                                   "x3_qb10_bias",
                                   "x3_qb15_bias",
                                   "x3_qb20_bias",
                                   "x3_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

x4_bias <- bias %>%
  select(starts_with("x4") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
                        levels = c("x4_uw_bias",
                                   "x4_ols_bias",
                                   "x4_cbgps_bias",
                                   "x4_npcbgps_bias",
                                   "x4_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

x5_bias <- bias %>%
  select(starts_with("x5") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
                        levels = c("x5_uw_bias",
                                   "x5_ols_bias",
                                   "x5_cbgps_bias",
                                   "x5_npcbgps_bias",
                                   "x5_olr_bias")),
         facet = str_sub(label, 1, 2),
         lab = str_sub(label, 4))

```

```

x6_bias <- bias %>%
  select(starts_with("x6") & ends_with("bias")) %>%
  gather(label, bias) %>%
  mutate(label = factor(label,
    levels = c("x6_uw_bias",
               "x6_ols_bias",
               "x6_cbgps_bias",
               "x6_npcbgps_bias",
               "x6_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, x4_bias, x5_bias, x6_bias) %>%
  mutate(facet = factor(facet, levels = c("x1", "x2", "x3", "x4", "x5", "x6")),
    lab = factor(lab, levels = c("uw_bias",
                                "ols_bias",
                                "cbgps_bias",
                                "npcbgps_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])),
                  expression(paste("D ", X[4])),
                  expression(paste("E ", X[5])),
                  expression(paste("F ", X[6])))))

# now plot with both austin and molr mse...
fig2_func <- function(exp_num, letter) {

  if(exp_num < 4) {
    mse_bias_plot %>% filter(facet == paste0("x", exp_num)) %>%
      ggplot(aes(y = fct_rev(lab), x = bias)) +
      stat_halfeye() +
      geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
      scale_y_discrete(name = "", labels = c("CPM", "QB20", "QB15", "QB10", "npCBGPS", "CBGPS", "OLS",
      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)) +
      annotate(geom = "text", x = 0.2, y = 9, label = "MSE", size = 10 / .pt) +
      annotate(geom = "text", x = 0.2, y = 8.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "UW") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
      annotate(geom = "text", x = 0.2, y = 7.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "OLS") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
      annotate(geom = "text", x = 0.2, y = 6.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "CBGPS") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
      annotate(geom = "text", x = 0.2, y = 5.5, label = mlor_mse %>%

```

```

        filter(Exposure == paste0("x", exp_num) & Method == "npCBGPS") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
    annotate(geom = "text", x = 0.2, y = 4.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "QB10") %>%
        pull(MSE) %>% round(4), size = 9 / .pt) +
    annotate(geom = "text", x = 0.2, y = 3.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "QB15") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
    annotate(geom = "text", x = 0.2, y = 2.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "QB20") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
    annotate(geom = "text", x = 0.2, y = 1.5, label = mlor_mse %>%
        filter(Exposure == paste0("x", exp_num) & Method == "OLR") %>%
        pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
    ggtitle(bquote(. (letter) * ") " ~ X[(exp_num)])) +
    coord_cartesian(clip = "off") +
    theme +
    theme(panel.grid.major.y = element_line(),
          axis.text = element_text(size = 8),
          axis.title = element_text(size = 9),
          legend.position = "none")
} else {
mse_bias_plot %>% filter(facet == paste0("x", exp_num)) %>%
  ggplot(aes(y = fct_rev(lab), x = bias)) +
  stat_halfeye() +
  geom_vline(xintercept = 0, alpha = 0.5, linetype = "dashed") +
  scale_y_discrete(name = "", labels = c("CPM", "npCBGPS", "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)) +
  annotate(geom = "text", x = 0.2, y = 6, label = "MSE", size = 10 / .pt) +
  annotate(geom = "text", x = 0.2, y = 5.5, label = mlor_mse %>%
    filter(Exposure == paste0("x", exp_num) & Method == "UW") %>%
    pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
  annotate(geom = "text", x = 0.2, y = 4.5, label = mlor_mse %>%
    filter(Exposure == paste0("x", exp_num) & Method == "OLS") %>%
    pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
  annotate(geom = "text", x = 0.2, y = 3.5, label = mlor_mse %>%
    filter(Exposure == paste0("x", exp_num) & Method == "CBGPS") %>%
    pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
  annotate(geom = "text", x = 0.2, y = 2.5, label = mlor_mse %>%
    filter(Exposure == paste0("x", exp_num) & Method == "npCBGPS") %>%
    pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
  annotate(geom = "text", x = 0.2, y = 1.5, label = mlor_mse %>%
    filter(Exposure == paste0("x", exp_num) & Method == "OLR") %>%
    pull(MSE) %>% round(4) %>% format(nsmall = 4), size = 9 / .pt) +
  ggtitle(bquote(. (letter) * ") " ~ X[(exp_num)])) +
  coord_cartesian(clip = "off") +
  theme +
  theme(panel.grid.major.y = element_line(),
        axis.text = element_text(size = 8),
        axis.title = element_text(size = 9),
        legend.position = "none")
}

```

```

}

ggarrange(fig2_func(1, "A"),
  fig2_func(2, "B"),
  fig2_func(3, "C"),
  fig2_func(4, "D"),
  fig2_func(5, "E"),
  fig2_func(6, "F"),
  ncol = 3,
  nrow = 2,
  heights = c(8, 5),
  align= "hv")

# save plot
ggsave("figs/fig2.pdf", width = 7, height = 7)
# embed the font
embed_fonts("figs/fig2.pdf")

```

## Mean Squared Error

Table 3 - Marginal Log Odds Ratio Approach Standard Error Ratio and Coverage

```

# now make table of mean squared error, which is the mean of the squared biases (or errors)
tab3 <- tibble(Method = c("Unweighted",
  "Standard Error Ratio",
  "Coverage",
  "Ordinary least squares",
  "Standard Error Ratio",
  "Coverage",
  "Covariate balancing generalized propensity score",
  "Standard Error Ratio",
  "Coverage",
  "Non-parametric covariate balancing generalized propensity score",
  "Standard Error Ratio",
  "Coverage",
  "Quantile binning categories",
  "10",
  "Standard Error Ratio",
  "Coverage",
  "15",
  "Standard Error Ratio",
  "Coverage",
  "20",
  "Standard Error Ratio",
  "Coverage",
  "Ordinal logistic regression",
  "Standard Error Ratio",
  "Coverage"),
  `X1` = c(NA,
    mean(bias$x1_uw_se) / sd(true_x1 - bias$x1_uw_bias),
    mean(bias$x1_uw_cov),
    NA,

```

```

mean(bias$x1_ols_se) / sd(true_x1 - bias$x1_ols_bias),
mean(bias$x1_ols_cov),
NA,
mean(bias$x1_cbgps_se) / sd(true_x1 - bias$x1_cbgps_bias),
mean(bias$x1_cbgps_cov),
NA,
mean(bias$x1_npcbgps_se) / sd(true_x1 - bias$x1_npcbgps_bias),
mean(bias$x1_npcbgps_cov),
NA,
NA,
mean(bias$x1_qb10_se) / sd(true_x1 - bias$x1_qb10_bias),
mean(bias$x1_qb10_cov),
NA,
mean(bias$x1_qb15_se) / sd(true_x1 - bias$x1_qb15_bias),
mean(bias$x1_qb15_cov),
NA,
mean(bias$x1_qb20_se) / sd(true_x1 - bias$x1_qb20_bias),
mean(bias$x1_qb20_cov),
NA,
mean(bias$x1_olr_se) / sd(true_x1 - bias$x1_olr_bias),
mean(bias$x1_olr_cov)),
`X4` = c(NA,
mean(bias$x4_uw_se) / sd(true_x4 - bias$x4_uw_bias),
mean(bias$x4_uw_cov),
NA,
mean(bias$x4_ols_se) / sd(true_x4 - bias$x4_ols_bias),
mean(bias$x4_ols_cov),
NA,
mean(bias$x4_cbgps_se) / sd(true_x4 - bias$x4_cbgps_bias),
mean(bias$x4_cbgps_cov),
NA,
mean(bias$x4_npcbgps_se) / sd(true_x4 - bias$x4_npcbgps_bias),
mean(bias$x4_npcbgps_cov),
NA,
NA, NA, NA,
NA, NA, NA,
NA, NA, NA,
NA,
mean(bias$x4_olr_se) / sd(true_x4 - bias$x4_olr_bias),
mean(bias$x4_olr_cov)),
`X2` = c(NA,
mean(bias$x2_uw_se) / sd(true_x2 - bias$x2_uw_bias),
mean(bias$x2_uw_cov),
NA,
mean(bias$x2_ols_se) / sd(true_x2 - bias$x2_ols_bias),
mean(bias$x2_ols_cov),
NA,
mean(bias$x2_cbgps_se) / sd(true_x2 - bias$x2_cbgps_bias),
mean(bias$x2_cbgps_cov),
NA,
mean(bias$x2_npcbgps_se) / sd(true_x2 - bias$x2_npcbgps_bias),
mean(bias$x2_npcbgps_cov),
NA,

```

```

NA,
mean(bias$x2_qb10_se) / sd(true_x2 - bias$x2_qb10_bias),
mean(bias$x2_qb10_cov),
NA,
mean(bias$x2_qb15_se) / sd(true_x2 - bias$x2_qb15_bias),
mean(bias$x2_qb15_cov),
NA,
mean(bias$x2_qb20_se) / sd(true_x2 - bias$x2_qb20_bias),
mean(bias$x2_qb20_cov),
NA,
mean(bias$x2_olr_se) / sd(true_x2 - bias$x2_olr_bias),
mean(bias$x2_olr_cov)),
`X5` = c(NA,
mean(bias$x5_uw_se) / sd(true_x5 - bias$x5_uw_bias),
mean(bias$x5_uw_cov),
NA,
mean(bias$x5_ols_se) / sd(true_x5 - bias$x5_ols_bias),
mean(bias$x5_ols_cov),
NA,
mean(bias$x5_cbgps_se) / sd(true_x5 - bias$x5_cbgps_bias),
mean(bias$x5_cbgps_cov),
NA,
mean(bias$x5_npcbgps_se) / sd(true_x5 - bias$x5_npcbgps_bias),
mean(bias$x5_npcbgps_cov),
NA,
NA, NA, NA,
NA, NA, NA,
NA, NA, NA,
NA,
mean(bias$x5_olr_se) / sd(true_x5 - bias$x5_olr_bias),
mean(bias$x5_olr_cov)),
`X3` = c(NA,
mean(bias$x3_uw_se) / sd(true_x3 - bias$x3_uw_bias),
mean(bias$x3_uw_cov),
NA,
mean(bias$x3_ols_se) / sd(true_x3 - bias$x3_ols_bias),
mean(bias$x3_ols_cov),
NA,
mean(bias$x3_cbgps_se) / sd(true_x3 - bias$x3_cbgps_bias),
mean(bias$x3_cbgps_cov),
NA,
mean(bias$x3_npcbgps_se) / sd(true_x3 - bias$x3_npcbgps_bias),
mean(bias$x3_npcbgps_cov),
NA,
NA,
mean(bias$x3_qb10_se) / sd(true_x3 - bias$x3_qb10_bias),
mean(bias$x3_qb10_cov),
NA,
mean(bias$x3_qb15_se) / sd(true_x3 - bias$x3_qb15_bias),
mean(bias$x3_qb15_cov),
NA,
mean(bias$x3_qb20_se) / sd(true_x3 - bias$x3_qb20_bias),
mean(bias$x3_qb20_cov),

```

```

      NA,
      mean(bias$x3_olr_se) / sd(true_x3 - bias$x3_olr_bias),
      mean(bias$x3_olr_cov)),
`X6` = c(NA,
      mean(bias$x6_uw_se) / sd(true_x6 - bias$x6_uw_bias),
      mean(bias$x6_uw_cov),
      NA,
      mean(bias$x6_ols_se) / sd(true_x6 - bias$x6_ols_bias),
      mean(bias$x6_ols_cov),
      NA,
      mean(bias$x6_cbgps_se) / sd(true_x6 - bias$x6_cbgps_bias),
      mean(bias$x6_cbgps_cov),
      NA,
      mean(bias$x6_npcbgps_se) / sd(true_x6 - bias$x6_npcbgps_bias),
      mean(bias$x6_npcbgps_cov),
      NA,
      NA, NA, NA,
      NA, NA, NA,
      NA, NA, NA,
      NA,
      mean(bias$x6_olr_se) / sd(true_x6 - bias$x6_olr_bias),
      mean(bias$x6_olr_cov))
)

# make table
kable(tab3, digits = 3) %>%
  kable_classic(html_font = "Arial", full_width = FALSE) %>%
  add_header_above(c("Marginally", "Normal" = 2, "Non-Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 2, "Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_indent(c(2:3, 5:6, 8:9, 11:12, 15:16, 18:19, 21:22, 24:25)) %>%
  add_indent(c(2:3, 5:6, 8:9, 11:12, 13:22, 24:25))

```

## Supplemental Table 2.5 - Marginal Log Odds Ratio Approach Bias

```

# now make table of biases
tab4 <- tibble(Method = c("Unweighted",
  "Ordinary least squares",
  "Covariate balancing generalized propensity score",
  "Non-parametric 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$x1_cbgps_bias, 0.75), 3), ")"),
paste0(round(median(bias$x1_npcbgps_bias), 4), " (",
        round(quantile(bias$x1_npcbgps_bias, 0.25), 3), ", ",
        round(quantile(bias$x1_npcbgps_bias, 0.75), 3), ")"),
NA,
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$x4_uw_bias), 4), " (",
        round(quantile(bias$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$x4_cbgps_bias, 0.75), 3), ")"),
        paste0(round(median(bias$x4_npcbgps_bias), 4), " (",
        round(quantile(bias$x4_npcbgps_bias, 0.25), 3), ", ",
        round(quantile(bias$x4_npcbgps_bias, 0.75), 3), ")"),
        NA,
        NA,
        NA,
        NA,
        paste0(round(median(bias$x4_olr_bias), 4), " (",
        round(quantile(bias$x4_olr_bias, 0.25), 3), ", ",
        round(quantile(bias$x4_olr_bias, 0.75), 3), ")"),
`Median Bias (IQR) ` = c(paste0(round(median(bias$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), ")"),
        paste0(round(median(bias$x2_npcbgps_bias), 4), " (",
        round(quantile(bias$x2_npcbgps_bias, 0.25), 3), ", ",
        round(quantile(bias$x2_npcbgps_bias, 0.75), 3), ")"),
        NA,
        paste0(round(median(bias$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$x5_uw_bias), 4), " (" ,
round(quantile(bias$x5_uw_bias, 0.25), 3), " , " ,
round(quantile(bias$x5_uw_bias, 0.75), 3), " ")",
paste0(round(median(bias$x5_ols_bias), 4), " (" ,
round(quantile(bias$x5_ols_bias, 0.25), 3), " , " ,
round(quantile(bias$x5_ols_bias, 0.75), 3), " ")",
paste0(round(median(bias$x5_cbgps_bias), 4), " (" ,
round(quantile(bias$x5_cbgps_bias, 0.25), 3), " , " ,
round(quantile(bias$x5_cbgps_bias, 0.75), 3), " ")",
paste0(round(median(bias$x5_npcbgps_bias), 4), " (" ,
round(quantile(bias$x5_npcbgps_bias, 0.25), 3), " , " ,
round(quantile(bias$x5_npcbgps_bias, 0.75), 3), " ")",
NA,
NA,
NA,
NA,
paste0(round(median(bias$x5_olr_bias), 4), " (" ,
round(quantile(bias$x5_olr_bias, 0.25), 3), " , " ,
round(quantile(bias$x5_olr_bias, 0.75), 3), " "))),
`Median Bias (IQR) ` = c(paste0(round(median(bias$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), " ")",
paste0(round(median(bias$x3_npcbgps_bias), 4), " (" ,
round(quantile(bias$x3_npcbgps_bias, 0.25), 3), " , " ,
round(quantile(bias$x3_npcbgps_bias, 0.75), 3), " ")",
NA,
paste0(round(median(bias$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), " ")",
paste0(round(median(bias$x3_olr_bias), 4), " (" ,
round(quantile(bias$x3_olr_bias, 0.25), 3), " , " ,

```

```

        round(quantile(bias$x3_olr_bias, 0.75), 3), "))),
`Median Bias (IQR)  ` = c(paste0(round(median(bias$x6_uw_bias), 4), " (",
        round(quantile(bias$x6_uw_bias, 0.25), 3), ", ",
        round(quantile(bias$x6_uw_bias, 0.75), 3), "))),
        paste0(round(median(bias$x6_ols_bias), 4), " (",
        round(quantile(bias$x6_ols_bias, 0.25), 3), ", ",
        round(quantile(bias$x6_ols_bias, 0.75), 3), "))),
        paste0(round(median(bias$x6_cbgps_bias), 4), " (",
        round(quantile(bias$x6_cbgps_bias, 0.25), 3), ", ",
        round(quantile(bias$x6_cbgps_bias, 0.75), 3), "))),
        paste0(round(median(bias$x6_npcbgps_bias), 4), " (",
        round(quantile(bias$x6_npcbgps_bias, 0.25), 3), ", ",
        round(quantile(bias$x6_npcbgps_bias, 0.75), 3), "))),
        NA,
        NA,
        NA,
        NA,
        paste0(round(median(bias$x6_olr_bias), 4), " (",
        round(quantile(bias$x6_olr_bias, 0.25), 3), ", ",
        round(quantile(bias$x6_olr_bias, 0.75), 3), ")))
    )

# make table
kable(tab4) %>%
  kable_classic(html_font = "Arial", full_width = TRUE) %>%
  add_header_above(c("", "X1" = 1, "X4" = 1, "X2" = 1,
    "X5" = 1, "X3" = 1, "X6" = 1), bold = TRUE) %>%
  add_header_above(c("Marginally", "Normal" = 2, "Non-Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_header_above(c("Conditionally", "Normal" = 2, "Normal" = 2,
    "Non-Normal" = 2), bold = TRUE) %>%
  add_indent(c(6:8))

```

Conditionally Marginally	Normal		Normal		Non-Normal	
	Normal		Non-Normal		Non-Normal	
	X1	X4	X2	X5	X3	X6
Method	Median Bias (IQR)	Median Bias (IQR)	Median Bias (IQR)	Median Bias (IQR)	Median Bias (IQR)	Median Bias (IQR)
Unweighted	-0.0613 (-0.107, -0.014)	-0.055 (-0.103, -0.01)	-0.0647 (-0.104, -0.029)	-0.0633 (-0.102, -0.026)	-0.0415 (-0.076, -0.009)	-0.0403 (-0.076, -0.006)
Ordinary least squares	-0.0024 (-0.055, 0.048)	-5e-04 (-0.054, 0.05)	-0.0118 (-0.077, 0.056)	-0.0507 (-0.089, -0.013)	-2e-04 (-0.04, 0.038)	-3e-04 (-0.039, 0.039)
Covariate balancing generalized propensity score	-0.0026 (-0.055, 0.049)	-1e-04 (-0.054, 0.05)	-0.0277 (-0.097, 0.054)	-0.0506 (-0.088, -0.013)	-8e-04 (-0.039, 0.038)	-4e-04 (-0.039, 0.038)
Non- parametric covariate balancing generalized propensity score	-8e-04 (-0.058, 0.056)	0.0023 (-0.054, 0.059)	-5e-04 (-0.071, 0.064)	-0.0496 (-0.087, -0.01)	0.01 (-0.034, 0.051)	0.0086 (-0.032, 0.049)
Quantile binning categories						
10	-0.009 (-0.059, 0.042)		-0.0207 (-0.074, 0.037)		-0.0084 (-0.046, 0.028)	
15	-0.0082 (-0.059, 0.043)		-0.0214 (-0.074, 0.036)		-0.0082 (-0.046, 0.029)	
20	-0.0077 (-0.058, 0.043)		-0.0209 (-0.075, 0.036)		-0.008 (-0.045, 0.029)	
Ordinal logistic regression	-0.0073 (-0.058, 0.044)	-0.0044 (-0.055, 0.045)	-0.0196 (-0.075, 0.035)	-0.049 (-0.087, -0.012)	-0.0079 (-0.045, 0.029)	-0.0072 (-0.044, 0.032)

## Session Info

```
sessionInfo()
```

```
R version 4.1.0 (2021-05-18)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS 12.6
```

```
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] grid parallel stats graphics grDevices utils datasets  
[8] methods base

other attached packages:

[1] survey\_4.1-1 ggdist\_3.0.0 extrafont\_0.17 doParallel\_1.0.16  
[5] iterators\_1.0.14 foreach\_1.5.2 furrr\_0.2.3 future\_1.23.0  
[9] ggpubr\_0.4.0 kableExtra\_1.3.4 lme4\_1.1-27.1 Matrix\_1.3-4  
[13] cobalt\_4.3.1 MatchThem\_1.0.1 WeightIt\_0.12.0 mice\_3.13.0  
[17] rms\_6.2-0 SparseM\_1.81 Hmisc\_4.6-0 Formula\_1.2-4  
[21] survival\_3.2-13 lattice\_0.20-45 forcats\_0.5.1 stringr\_1.4.0  
[25] dplyr\_1.0.8 purrr\_0.3.4 readr\_2.0.2 tidyr\_1.2.0  
[29] tibble\_3.1.6 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 prodlim\_2019.11.13 htmlwidgets\_1.5.4  
[55] httr\_1.4.2 MatchIt\_4.3.0 RColorBrewer\_1.1-2  
[58] ellipsis\_0.3.2 farver\_2.1.0 pkgconfig\_2.0.3  
[61] nnet\_7.3-16 dbplyr\_2.1.1 utf8\_1.2.2  
[64] caret\_6.0-90 labeling\_0.4.2 tidyselect\_1.1.2  
[67] rlang\_1.0.2 reshape2\_1.4.4 munsell\_0.5.0  
[70] cellranger\_1.1.0 tools\_4.1.0 cli\_3.2.0  
[73] moments\_0.14 generics\_0.1.2 broom\_0.7.10  
[76] evaluate\_0.15 fastmap\_1.1.0 yaml\_2.3.5  
[79] ModelMetrics\_1.2.2.2 knitr\_1.37 fs\_1.5.2  
[82] nlme\_3.1-153 quantreg\_5.86 xml2\_1.3.3  
[85] compiler\_4.1.0 rstudioapi\_0.13 png\_0.1-7  
[88] ggsignif\_0.6.3 reprex\_2.0.1 stringi\_1.7.6  
[91] nloptr\_1.2.2.3 vctrs\_0.3.8 pillar\_1.7.0  
[94] lifecycle\_1.0.1 cowplot\_1.1.1 data.table\_1.14.2  
[97] conquer\_1.2.1 R6\_2.5.1 latticeExtra\_0.6-29  
[100] gridExtra\_2.3 parallelly\_1.28.1 codetools\_0.2-18

[103]	polyspline_1.1.19	boot_1.3-28	MASS_7.3-54
[106]	assertthat_0.2.1	withr_2.5.0	multcomp_1.4-17
[109]	hms_1.1.1	rpart_4.1-15	timeDate_3043.102
[112]	class_7.3-19	minqa_1.2.4	rmarkdown_2.13
[115]	carData_3.0-4	pROC_1.18.0	lubridate_1.8.0
[118]	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.