casual_project_first_run

Tiankai

2023-11-17

First, load in necessary functions for this project.

```
# 2018-2020 NHANES proportion (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.025558
# 2011-2018 NHANES proportion (https://jamanetwork.com/journals/jama/fullarticle/2784659)

library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(dplyr)

#### ---- Define Custom Functions
simulate_data <- function(k) {

  # Parameters for sample sizes
  N = 5000
  N_Trial = floor(0.5*N)
  N_Target = N - N_Trial

  # Parameters for differential specification
  l <- 0.5
  p <- 0.5

  ##### Gender #####
  p_male1 = 0.396 # p_male = 0.396 from NHANES
  Male_trial <- rbinom(N_Trial, 1, p_male1)
  p_male2 = 0.492
  Male_target <- rbinom(N_Target, 1, p_male2)

  Male = c(Male_trial, Male_target)

  ##### Age #####
```

```r
# Trial
alpha <- 7.5
beta <- 5.4
Sim_age_Trial <- rbeta(N_Trial, alpha, beta)

# Adjust the simulated age data to your desired age range
min_age <- 15
max_age <- 80
Sim_age_Trial <- min_age + Sim_age_Trial * (max_age - min_age)

# Target
alpha <- 6
beta <- 5
Sim_age_Target <- rbeta(N_Target, alpha, beta)

# Adjust the simulated age data to your desired age range
min_age <- 18
max_age <- 75
Sim_age_Target <- min_age + Sim_age_Target * (max_age - min_age)

Age = c(Sim_age_Trial, Sim_age_Target)

#Race
Race_Trial <-sample(c("Non-Hispanic White", "Non-Hispanic Black", "Hispanic", "Asian", "Other"),
                    N_Trial, replace=TRUE, prob=c(0.508, 0.234, 0.205, 0.032, 0.021)) # From NHANES
Race_Target <-sample(c("Non-Hispanic White", "Non-Hispanic Black", "Hispanic", "Asian", "Other"),
                     N_Target, replace=TRUE, prob=c(0.381, 0.235, 0.245, 0.131, 0.008)) # From CVD st
Race <- as.factor(c(Race_Trial, Race_Target))

# Use model.matrix to create one-hot encoded vectors
# The "-1" removes the intercept term
Race_one_hot_encoded <- model.matrix(~ Race - 1, data.frame(Race)) %>% as.data.frame()

# Rename the columns for clarity (optional)
colnames(Race_one_hot_encoded) <- levels(Race)

#BMI
slope_age <- 0.6 # From NHANES
slope_sex <- -0.425 # From NHANES
BMI <- slope_age * Age + slope_sex * Male  + rnorm(N, mean = 0, sd = 6.5) # Males on average have low

# Generate Latent Variable
UTrial <- rnorm(N_Trial)
UTarget <- rnorm(N_Target)

# Cut Latent Variable
Trial_cut_less <- rbinom(1, 1, prob = 0.5) # Binary indicator for Trial cut point less than Target cu
a <- l - k/2
b <- l + k/2
a_thresh <- qnorm(a)
b_thresh <- qnorm(b)
if (Trial_cut_less==1) {
  VTrial <- ifelse(UTrial < a_thresh, "Low", "High")
```

```
    VTarget <- ifelse(UTarget < b_thresh, "Low", "High")
  } else {
    VTrial <- ifelse(UTrial < b_thresh, "Low", "High")
    VTarget <- ifelse(UTarget < a_thresh, "Low", "High")
  }

  V = factor(c(VTrial, VTarget), levels = c("Low", "High"))

  # Create treatment and study indicator variables
  ATrial <- rbinom(N_Trial, 1, .5)
  A <- c(ATrial, rep(NA, N_Target)) # 1 if treat, 0 if control (only for those in trial!)
  S <- c(rep(1, N_Trial), rep(0, N_Target)) # indicator variable for trial

  # Define potential outcomes
  ATE <- 30
  beta_0 <- 100
  epsilon <- 50
  beta_race <- matrix(5:1, ncol = 1)
  beta_male <- 3
  beta_age <- -.2
  beta_bmi <- 1
  beta_V <- 3

  effect_race <- as.matrix(Race_one_hot_encoded) %*% beta_race %>%
    as.vector()

  Y0 <- beta_male * Male + beta_age * Age + beta_bmi * BMI + beta_V * as.numeric(V=="High") +
    effect_race + rnorm(N, beta_0, sd = epsilon)
  Y1 <- beta_male * Male + beta_age * Age + beta_bmi * BMI + beta_V * as.numeric(V=="High") +
    effect_race + rnorm(N, beta_0, sd = epsilon) + ATE
  Y <- ifelse(A==1, Y1, Y0) # Y is NA for subjects in the target population
  data <- data.frame(Male, Age, Race, BMI, V, A, S, Y0, Y1, Y)

  ## -- Quality Assure Data
  # Check to ensure that after grouping by S and A, each Race level includes at least one observation
  N_Race <- levels(Race) %>% length()
  N_Race_by_group <- data %>% group_by(S, A) %>% count(Race) %>% group_split() %>% sapply(nrow)
  if (any(N_Race_by_group != N_Race)) { stop("After grouping by S and A, at least one level of Race has

  return(data)
}

get_tate_true <- function(data) {
  targetdata <- subset(data, S==0)
  tateTrue <- mean(targetdata$Y1-targetdata$Y0)
  return(tateTrue)
}

estimate_tates <- function(data) {

  # Make A and S available
  attach(data, warn.conflicts = FALSE)
```

```r
# Outcome regression version
# Step one: Create necessary datasets
studydata <- subset(data, S==1)
studydataT <- subset(studydata, A==1)
studydataC <- subset(studydata, A==0)
targetdata <- subset(data, S==0)

# Step two: Make models for E[Y|X, S=1, A=a]
treatlm <- lm(Y ~ Male+Age+Race+BMI+V, data=studydataT)
contlm <- lm(Y ~ Male+Age+Race+BMI+V, data=studydataC)

#Step 3: Get predicted values for set st S=0 for both assignments
# set A=1
targetdata$A <- 1
gt <- predict(treatlm, newdata = targetdata) # predict treatlm on targetdata
# set A=0
targetdata$A <- 0
gc <- predict(contlm, newdata = targetdata) # predict contlm on targetdata

#Step 4: get E[Y^a|S=0]'s
n_target <- nrow(targetdata)
mut <- sum(gt) / n_target
muc <- sum(gc) / n_target

#Step 5: subtract them to get the estimated TATE
tateOR <- mut - muc

#IOW1
#set up weights - there is one set per treatment
p <- 1/(1+exp(-1* predict(glm(S ~ Male+Age+Race+BMI+V, data=data, family="binomial")))) #P(S=1|X)
p <- p[S==1] # Only study data matters for estimation because of indicator function
e1 <- 1/(1+exp(-1* predict(glm(A ~ Male+Age+Race+BMI+V, data=studydata, family="binomial")))) # P(A=1
e0 <- 1/(1+exp(-1*predict(glm((1-A) ~ Male+Age+Race+BMI+V, data=studydata, family="binomial")))) # P(
w1 <- (1-p)/(p*e1)
w0 <- (1-p)/(p*e0)

study_treated_ids <- which(studydata$A==1)
w1 <- w1[study_treated_ids]
w0 <- w0[-study_treated_ids]
study_y1 <- studydata$Y[study_treated_ids]
study_y0 <- studydata$Y[-study_treated_ids]

muIOW1t <- sum(w1 * study_y1) / n_target
muIOW1c <- sum(w0 * study_y0) / n_target
tateIOW1 <- muIOW1t-muIOW1c

#IOW2
muIOW2t <- sum(w1 * study_y1) / sum(w1)
muIOW2c <- sum(w0 * study_y0) / sum(w0)
tateIOW2 <- muIOW2t-muIOW2c

#DR1
study_gt <- predict(treatlm, newdata = studydataT)
```

```r
  study_part_t <- sum(w1*(study_y1 - study_gt)) # Specific to those in study and received treatment
  target_part_t <- sum(gt) # Specific to target and received treatment
  muDR1t <- (study_part_t + target_part_t) / n_target

  study_gc <- predict(treatlm, newdata = studydataC)
  study_part_c <- sum(w0*(study_y0 - study_gc)) # Specific to those in study and did not receive treatm
  target_part_c <- sum(gc) # Specific to those in target and did not receive treatment
  muDR1c <- (study_part_c + target_part_c) / n_target

  tateDR1 <- muDR1t - muDR1c

  #DR2
  # TODO ensure robustness to bootstrap data
  muDR2t <- study_part_t / sum(w1) + target_part_t / n_target
  muDR2c <- study_part_c / sum(w0) + target_part_c / n_target
  tateDR2 <- muDR2t - muDR2c

  tates <- c(tateOR, tateIOW1, tateIOW2, tateDR1, tateDR2)
  return(tates)
}

# Function to perform bootstrap resampling and estimate the five quantities
bootstrap_tates <- function(data, B=100) {

  # Number of observations
  n_obs <- nrow(data)

  # Number of estimates
  num_estimates <- 5

  # Matrix to store bootstrap estimates
  bootstrap_matrix <- matrix(NA, nrow = B, ncol = num_estimates)

  # Perform bootstrap resampling
  for (b in 1:B) {

    # Sample with replacement
    bootstrap_ids <- sample(1:n_obs, n_obs, replace = TRUE)
    bootstrap_sample <- data[bootstrap_ids, ]

    # Calculate the five estimates
    bootstrap_estimates <- estimate_tates(bootstrap_sample)

    # Store the estimates in the matrix
    bootstrap_matrix[b, ] <- bootstrap_estimates
  }

  return(bootstrap_matrix)
}

# Determine for checking if the truth is in the confidence intervals
check_inclusion <- function(ci, truth) {
  truth_included <- as.numeric(ci[1] < truth & truth < ci[2])
  return (truth_included)
}
```

ANALYSIS

Then we try to run under different k.

```
set.seed(343)

k <- 0.05

# Number of estimates
num_estimates <- 5

# Matrix for coverage indicators
M <- 10

true_tates <- rep(NA, M)
tate_matrix <- coverage_matrix <- matrix(NA, nrow = M, ncol = num_estimates)

for (m in 1:M) {

  # Simulate
  data_m <- simulate_data(k)
  # Get the truth
  tate_true_m <- get_tate_true(data_m)
  # Point estimate tates
  tates_m <- estimate_tates(data_m)
  # Bootstrap variability
  bootstrap_matrix <- bootstrap_tates(data_m)
  # Get Confidence Intervals
  confidence_intervals <- apply(bootstrap_matrix, MARGIN = 2,
                               FUN = quantile, probs = c(0.025, 0.975))
  # Determine if the truth is in the confidence intervals
  coverage_indicators <- apply(confidence_intervals, MARGIN = 2,
                              check_inclusion, tate_true_m)

  # Store outputs
  true_tates[m] <- tate_true_m
  tate_matrix[m, ] <- tates_m
  coverage_matrix[m, ] <- coverage_indicators
}

result_k0.05 <- list(true_tates, tate_matrix, coverage_matrix)
names(result_k0.05) <- c("true_tates", "tate_matrix", "coverage_matrix")
```

Display result at k=0.05

```
result_k0.05
```

```
## $true_tates
##  [1] 31.22494 29.67187 30.47004 27.75533 32.29068 28.94657 30.31125 27.73116
##  [9] 30.64104 27.77071
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 31.68444 31.97033 32.00098 63.84060 63.46283
##  [2,] 24.81469 22.96737 25.66782 50.46459 50.27840
##  [3,] 32.19614 33.56055 32.86480 65.02552 65.00056
##  [4,] 25.34541 29.59906 25.70164 50.74605 50.93392
##  [5,] 32.51474 33.83083 31.37841 63.97800 64.04473
##  [6,] 31.14577 32.05404 31.39118 62.47213 62.68461
##  [7,] 28.49184 30.53317 28.01842 56.66546 56.89485
##  [8,] 34.30116 37.12563 34.48226 68.58527 68.97561
##  [9,] 31.46560 31.19234 31.47891 63.63796 63.24781
## [10,] 28.97516 30.64378 28.88230 57.91916 58.27858
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    0    0
##  [2,]    0    0    1    0    0
##  [3,]    1    1    1    0    0
##  [4,]    1    1    1    0    0
##  [5,]    1    1    1    0    0
##  [6,]    1    1    1    0    0
##  [7,]    1    1    1    0    0
##  [8,]    0    0    0    0    0
##  [9,]    1    1    1    0    0
## [10,]    1    1    1    0    0
```

Create average value for the results

```
Truth_W_tate = cbind.data.frame(truth = as.matrix(result_k0.05$true_tates,nrow=M),result_k0.05$tate_mat
Avg_truth_W_tate_k0.05 = colMeans(Truth_W_tate)
Avg_truth_W_tate_k0.05
```

```
##    truth        1        2        3        4        5
## 29.68136 30.09349 31.34771 30.18667 60.33347 60.38019
```

```
Bias = result_k0.05$tate_matrix - result_k0.05$true_tates
Avg_bias_k0.05 = colMeans(Bias)
Avg_bias_k0.05
```

```
## [1]  0.4121355  1.6663502  0.5053149 30.6521157 30.6988305
```

```
Avg_cov_k0.05 = colMeans(result_k0.05$coverage_matrix)
Avg_cov_k0.05
```

```
## [1] 0.8 0.8 0.9 0.0 0.0
```

```
Avg_res_k0.05 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.05,
                     Avg_bias = Avg_bias_k0.05, Avg_cov = Avg_cov_k0.05)
```

I also tried a different k

```r
set.seed(343)

k <- 0.2

# Number of estimates
num_estimates <- 5

# Matrix for coverage indicators
M <- 10

true_tates <- rep(NA, M)
tate_matrix <- coverage_matrix <- matrix(NA, nrow = M, ncol = num_estimates)

for (m in 1:M) {

  # Simulate
  data_m <- simulate_data(k)
  # Get the truth
  tate_true_m <- get_tate_true(data_m)
  # Point estimate tates
  tates_m <- estimate_tates(data_m)
  # Bootstrap variability
  bootstrap_matrix <- bootstrap_tates(data_m)
  # Get Confidence Intervals
  confidence_intervals <- apply(bootstrap_matrix, MARGIN = 2,
                                FUN = quantile, probs = c(0.025, 0.975))
  # Determine if the truth is in the confidence intervals
  coverage_indicators <- apply(confidence_intervals, MARGIN = 2,
                               check_inclusion, tate_true_m)

  # Store outputs
  true_tates[m] <- tate_true_m
  tate_matrix[m, ] <- tates_m
  coverage_matrix[m, ] <- coverage_indicators
}

result_k0.2 <- list(true_tates, tate_matrix, coverage_matrix)
names(result_k0.2) <- c("true_tates", "tate_matrix", "coverage_matrix")
```

Display result at k=0.2

```r
result_k0.2
```

```
## $true_tates
##  [1] 31.22494 29.67187 30.47004 27.75533 32.29068 28.94657 30.31125 27.73116
##  [9] 30.64104 27.77071
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 32.63923 31.30821 32.40304 65.71064 65.15707
##  [2,] 23.87201 19.50438 25.30881 49.43810 48.95323
##  [3,] 32.69073 35.54551 34.41392 67.02386 67.06483
##  [4,] 25.20136 29.74857 25.23160 49.83566 50.34544
##  [5,] 32.43531 30.09898 29.70795 62.99584 62.75821
##  [6,] 31.24634 32.61083 31.08026 62.20528 62.66234
##  [7,] 28.42574 32.84336 27.89892 56.55615 56.83525
##  [8,] 34.94491 36.34191 35.65438 70.83417 70.92971
##  [9,] 30.85631 28.81091 30.58932 62.17712 61.67022
## [10,] 28.90802 30.65918 28.96899 57.91762 58.28785
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    0    0
##  [2,]    0    0    1    0    0
##  [3,]    1    1    1    0    0
##  [4,]    1    1    1    0    0
##  [5,]    1    1    1    0    0
##  [6,]    1    1    1    0    0
##  [7,]    1    1    1    0    0
##  [8,]    0    0    0    0    0
##  [9,]    1    1    1    0    0
## [10,]    1    1    1    0    0
```

Create average value for the results above

```
Truth_W_tate = cbind.data.frame(truth = as.matrix(result_k0.2$true_tates,nrow=M),result_k0.2$tate_matri
Avg_truth_W_tate_k0.2 = colMeans(Truth_W_tate)
Avg_truth_W_tate_k0.2
```

```
##    truth        1        2        3        4        5
## 29.68136 30.12200 30.74718 30.12572 60.46944 60.46642
```

```
Bias = result_k0.2$tate_matrix - result_k0.2$true_tates
Avg_bias_k0.2 = colMeans(Bias)
Avg_bias_k0.2
```

```
## [1]  0.4406365  1.0658239  0.4443592 30.7880835 30.7850573
```

```
Avg_cov_k0.2 = colMeans(result_k0.2$coverage_matrix)
Avg_cov_k0.2
```

```
## [1] 0.8 0.8 0.9 0.0 0.0
```

```
Avg_res_k0.2 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.2,
                    Avg_bias = Avg_bias_k0.2, Avg_cov = Avg_cov_k0.2)
```

```
set.seed(343)

k <- 0.8

# Number of estimates
num_estimates <- 5
```

```r
# Matrix for coverage indicators
M <- 10

true_tates <- rep(NA, M)
tate_matrix <- coverage_matrix <- matrix(NA, nrow = M, ncol = num_estimates)

for (m in 1:M) {

  # Simulate
  data_m <- simulate_data(k)
  # Get the truth
  tate_true_m <- get_tate_true(data_m)
  # Point estimate tates
  tates_m <- estimate_tates(data_m)
  # Bootstrap variability
  bootstrap_matrix <- bootstrap_tates(data_m)
  # Get Confidence Intervals
  confidence_intervals <- apply(bootstrap_matrix, MARGIN = 2,
                                FUN = quantile, probs = c(0.025, 0.975))
  # Determine if the truth is in the confidence intervals
  coverage_indicators <- apply(confidence_intervals, MARGIN = 2,
                               check_inclusion, tate_true_m)

  # Store outputs
  true_tates[m] <- tate_true_m
  tate_matrix[m, ] <- tates_m
  coverage_matrix[m, ] <- coverage_indicators
}

result_k0.8 <- list(true_tates, tate_matrix, coverage_matrix)
names(result_k0.8) <- c("true_tates", "tate_matrix", "coverage_matrix")
```

```r
result_k0.8
```

```
## $true_tates
##  [1] 31.22494 29.67187 30.47004 27.75533 32.29068 28.94657 30.31125 27.73116
##  [9] 30.64104 27.77071
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 36.31594 49.31487 29.62565 66.20436 67.27026
##  [2,] 29.47321 34.77808 31.90920 62.91455 61.83916
##  [3,] 36.83888 15.21277 40.67205 77.32236 74.33431
##  [4,] 26.96256 22.95181 27.96462 53.66437 53.44494
##  [5,] 29.71495  2.65881 18.47615 48.96923 47.71370
##  [6,] 27.66797 19.06975 30.92535 59.82929 58.70527
##  [7,] 36.32481 29.86347 29.34751 66.12992 66.79468
##  [8,] 37.02813 37.15538 41.63376 79.28136 80.13721
##  [9,] 21.39102 41.45809 35.36586 51.86140 51.37524
## [10,] 35.79874 45.66187 40.56853 77.79132 77.15101
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    0    0
##  [2,]    1    1    1    0    0
##  [3,]    1    1    1    0    0
##  [4,]    1    1    1    0    0
##  [5,]    1    1    1    1    1
##  [6,]    1    1    1    0    0
##  [7,]    1    1    1    0    0
##  [8,]    1    1    1    0    0
##  [9,]    1    1    1    1    1
## [10,]    1    1    1    0    0
```

```
Truth_W_tate = cbind.data.frame(truth = as.matrix(result_k0.8$true_tates,nrow=M),result_k0.8$tate_matri
Avg_truth_W_tate_k0.8 = colMeans(Truth_W_tate)
Avg_truth_W_tate_k0.8
```

```
##    truth        1        2        3        4        5
## 29.68136 31.75162 29.81249 32.64887 64.39681 63.87658
```

```
Bias = result_k0.8$tate_matrix - result_k0.8$true_tates
Avg_bias_k0.8 = colMeans(Bias)
Avg_bias_k0.8
```

```
## [1]  2.0702639  0.1311315  2.9675074 34.7154555 34.1952185
```

```
Avg_cov_k0.8 = colMeans(result_k0.8$coverage_matrix)
Avg_cov_k0.8
```

```
## [1] 1.0 1.0 1.0 0.2 0.2
```

```
Avg_res_k0.8 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.8,
                    Avg_bias = Avg_bias_k0.8, Avg_cov = Avg_cov_k0.8)
```

RESULT

```
res2vec = function(avg_res){
  dt = matrix(c(as.numeric(avg_res$Avg_truth_W_tate[-1]),
                avg_res$Avg_bias,avg_res$Avg_cov), ncol=3,nrow = 5)
vec = round(c(as.numeric(avg_res$Avg_truth_W_tate[1]),c(t(dt))),3)
return(vec)
}

avg_vec_k0.05 = res2vec(Avg_res_k0.05)
avg_vec_k0.2 = res2vec(Avg_res_k0.2)
avg_vec_k0.8 = res2vec(Avg_res_k0.8)
```

```
avg_tb = rbind.data.frame(avg_vec_k0.05,avg_vec_k0.2,avg_vec_k0.8)

colnames(avg_tb) = c("Truth","ATE_OR","bias_OR","cov_OR","ATE_IPW","bias_IPW",
```

```r
                      "cov_IPW","ATE_IPW2","bias_IPW2","cov_IPW2","ATE_DRE1",
                      "bias_DRE1","cov_DRE1","ATE_DRE2","bias_DRE2","cov_DRE2")

rownames(avg_tb) = c("K=0.05","K=0.2","K=0.8")

avg_tb
```

```
##          Truth ATE_OR bias_OR cov_OR ATE_IPW bias_IPW cov_IPW ATE_IPW2 bias_IPW2
## K=0.05 29.681 30.093   0.412    0.8  31.348    1.666     0.8   30.187     0.505
## K=0.2  29.681 30.122   0.441    0.8  30.747    1.066     0.8   30.126     0.444
## K=0.8  29.681 31.752   2.070    1.0  29.812    0.131     1.0   32.649     2.968
##        cov_IPW2 ATE_DRE1 bias_DRE1 cov_DRE1 ATE_DRE2 bias_DRE2 cov_DRE2
## K=0.05      0.9   60.333    30.652      0.0   60.380    30.699      0.0
## K=0.2       0.9   60.469    30.788      0.0   60.466    30.785      0.0
## K=0.8       1.0   64.397    34.715      0.2   63.877    34.195      0.2
```

{r} # library(xtable) # print(xtable(avg_tb, type = "latex"))

Seperate tables

```r
ATE_sub = avg_tb %>% select(Truth,ATE_OR,ATE_IPW,ATE_IPW2,ATE_DRE1,ATE_DRE2)
ATE_sub
```

```
##          Truth ATE_OR ATE_IPW ATE_IPW2 ATE_DRE1 ATE_DRE2
## K=0.05 29.681 30.093  31.348   30.187   60.333   60.380
## K=0.2  29.681 30.122  30.747   30.126   60.469   60.466
## K=0.8  29.681 31.752  29.812   32.649   64.397   63.877
```

```r
bias_sub = avg_tb %>% select(bias_OR,bias_IPW,bias_IPW2,bias_DRE1,bias_DRE2)
bias_sub
```

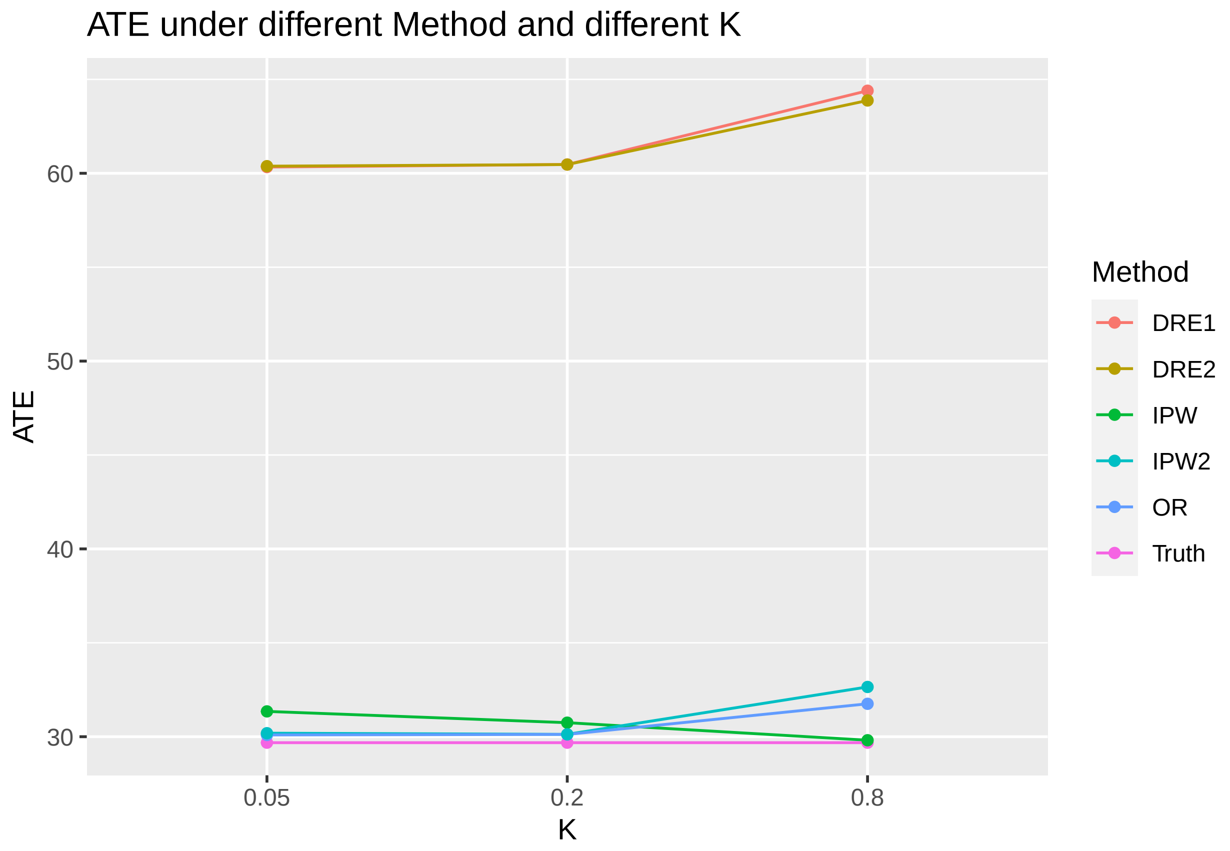
```
##        bias_OR bias_IPW bias_IPW2 bias_DRE1 bias_DRE2
## K=0.05   0.412    1.666     0.505    30.652    30.699
## K=0.2    0.441    1.066     0.444    30.788    30.785
## K=0.8    2.070    0.131     2.968    34.715    34.195
```

```r
cov_sub = avg_tb %>% select(cov_OR,cov_IPW,cov_IPW2,cov_DRE1,cov_DRE2)
cov_sub
```

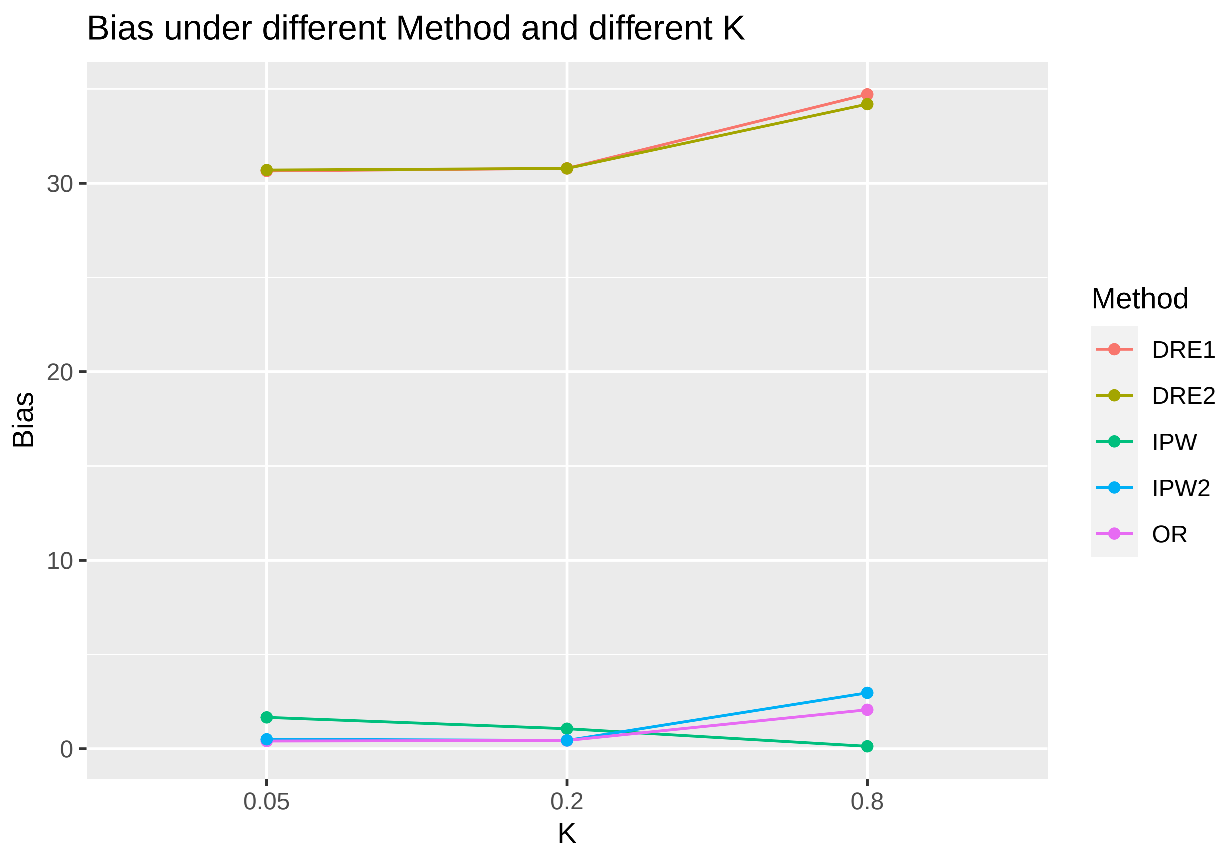
```
##        cov_OR cov_IPW cov_IPW2 cov_DRE1 cov_DRE2
## K=0.05    0.8     0.8      0.9      0.0      0.0
## K=0.2     0.8     0.8      0.9      0.0      0.0
## K=0.8     1.0     1.0      1.0      0.2      0.2
```

Here is for graphing

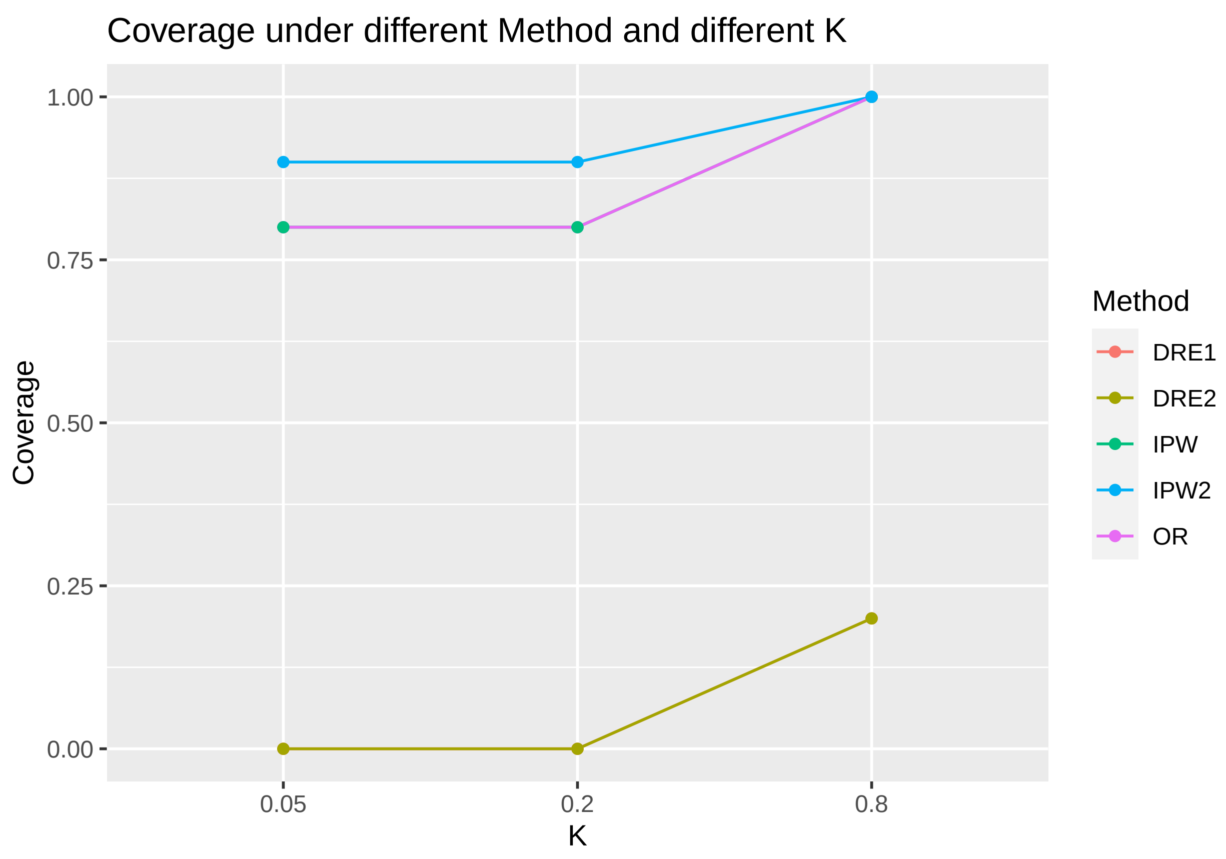
```r
k_lst = c("0.05","0.2","0.8")

##### ATE #####
method = rep(c("Truth","OR" ,"IPW" ,"IPW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,6)
ATE_lst = as.numeric(unlist(ATE_sub))
ATE_tb = cbind.data.frame(Method = method,K = k,ATE = ATE_lst)

library(ggplot2)
ggplot(data=ATE_tb, aes(x=K, y=ATE_lst, group=Method)) +
  geom_line(aes(color=Method))+
  geom_point(aes(color=Method))+
  xlab("K") +
  ylab("ATE") +
  ggtitle("ATE under different Method and different K")
```



```r
##### Bias #####
method = rep(c("OR" ,"IPW" ,"IPW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,5)
bias_lst = as.numeric(unlist(bias_sub))
bias_tb = cbind.data.frame(Method = method,K = k,bias = bias_lst)

library(ggplot2)
ggplot(data=bias_tb, aes(x=K, y=bias_lst, group=Method)) +
  geom_line(aes(color=Method))+
  geom_point(aes(color=Method))+
  xlab("K") +
  ylab("Bias") +
  ggtitle("Bias under different Method and different K")
```



```
##### Coverage #####
method = rep(c("OR" ,"IPW" ,"IPW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,5)
cov_lst = as.numeric(unlist(cov_sub))
cov_tb = cbind.data.frame(Method = method,K = k,cov = cov_lst)

library(ggplot2)
ggplot(data=cov_tb, aes(x=K, y=cov_lst, group=Method)) +
  geom_line(aes(color=Method))+
  geom_point(aes(color=Method))+
  xlab("K") +
  ylab("Coverage") +
  ggtitle("Coverage under different Method and different K")
```


Coverage under different Method and different K
Coverage
1.00
0.75
0.50
0.25
0.00
0.05
0.2
0.8
K
Method
DRE1
DRE2
IPW
IPW2
OR