casual_project_first_run

Tiankai

2023-11-17

First, load in necessary functions for this project.

```r
# 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
```

```r
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 #####
```

```
# 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 # fix Asian+other
Race_Trial <-sample(c("Non-Hispanic White", "Non-Hispanic Black", "Hispanic", "Other"),
                    N_Trial, replace=TRUE, prob=c(0.518, 0.239, 0.209, 0.034)) # From Obesity NHANES
Race_Target <-sample(c("Non-Hispanic White", "Non-Hispanic Black", "Hispanic", "Other"),
                     N_Target, replace=TRUE, prob=c(0.382, 0.237, 0.247, 0.134)) # From CVD NHANES
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)
U = c(UTrial, UTarget)

# 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 # l=0.5
b <- l + k/2
a_thresh <- qnorm(a)
b_thresh <- qnorm(b)
if (Trial_cut_less==1) {
```

```r
    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_param = 5
  ATE <- 30 + U*ATE_param
  beta_0 <- 100
  epsilon <- 50
  beta_race <- matrix(4:1, 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) {
  S<-data$S
  A<-data$A
```

```
#Analyze data
  #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$A <- 1
        gt<-data$gt <- predict(treatlm,newdata =data)
      #set A=0
        targetdata$A <- 0
        gc<-data$gc <- predict(contlm, newdata = data)

    #Step 4: get E[Y^a|S=0]'s
    mut <- (1/sum((1-S)))*sum((1-S)*gt)
    muc <- (1/sum((1-S)))*sum((1-S)*gc)
    #Step 5: subtract them to get the estimated TATE
    tateOR <- mut - muc


  #IOW1
    data$p <- 1/(1+exp(-1* predict(glm(S ~ Male+Age+Race+BMI+V, data=data, family="binomial"),newdata =
    data$e1 <- 1/(1+exp(-1* predict(glm(A ~ Male+Age+Race+BMI+V, data=studydata, family="binomial"),new
    data$e0 <-  1/(1+exp(-1*predict(glm((1-A) ~ Male+Age+Race+BMI+V, data=studydata, family="binomial")

    data$w1<-ifelse(S==1 & A==1, (1-data$p)/(data$p*data$e1), 0)
    data$w0<-ifelse(S==1 & A==0, (1-data$p)/(data$p*data$e0), 0)
    muIOW1t <- (1/sum((1-S)))*sum(data$w1*data$Y,na.rm =T)
    muIOW1c <- (1/sum((1-S)))*sum(data$w0*data$Y,na.rm =T)
    tateIOW1 <- muIOW1t-muIOW1c

   #IOW2
    muIOW2t <- (1/sum(data$w1,na.rm =T))*sum(data$w1*data$Y,na.rm =T)
    muIOW2c <- (1/sum(data$w0,na.rm =T))*sum(data$w0*data$Y,na.rm =T)
    tateIOW2 <- muIOW2t-muIOW2c

  #DR1
    muDR1t <-  (1/sum((1-S)))*(sum(data$w1*(data$Y-gt),na.rm =T)+sum((1-S)*gt,na.rm =T))
    muDR1c <- (1/sum((1-S)))*(sum(data$w0*(data$Y-gc),na.rm =T) +sum((1-S)*gc,na.rm =T))
    tateDR1 <- muDR1t - muDR1c

  #DR2
    muDR2t <-  (1/sum(data$w1,na.rm =T))*sum(data$w1*(data$Y-gt),na.rm =T) +(1/sum((1-S)))*sum((1-S)*gt
    muDR2c <-  (1/sum(data$w0,na.rm =T))*sum(data$w0*(data$Y-gc),na.rm =T) +(1/sum((1-S)))*sum((1-S)*gc
    tateDR2 <- muDR2t - muDR2c
```

```r
  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

Here is the base case.

```r
k <- 0

# Matrix for coverage indicators
M <- 50

seed = floor(20*k*M+343)

set.seed(seed)
```

```r
# Number of estimates
num_estimates <- 5

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

Display result at k=0

```r
result_k0
```

```
## $true_tates
##  [1] 31.33424 29.56007 30.41784 27.85470 32.16864 28.95896 30.22323 27.71503
##  [9] 30.77214 27.71291 32.41418 30.15274 32.63579 32.86611 28.44933 31.76586
## [17] 30.26036 29.15393 30.36893 30.61119 30.35885 31.73217 30.68192 28.63535
## [25] 30.35921 28.81267 29.28284 27.12628 30.12978 30.96399 30.63117 28.85089
## [33] 30.10134 29.13999 28.59707 30.11770 29.95304 29.88697 30.23062 27.46545
## [41] 28.40236 30.88260 33.86053 30.13953 30.32009 28.93696 28.65510 30.47754
## [49] 27.47460 28.84167
##
## $tate_matrix
##          [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 30.55600 30.27586 28.63655 28.40127 28.43548
##  [2,] 26.78764 26.00760 26.79326 27.13042 27.12307
##  [3,] 30.21649 26.67123 30.98937 30.19225 30.19531
##  [4,] 25.38845 29.07763 25.13639 25.56995 25.56890
##  [5,] 32.84327 34.81232 32.71685 32.81250 32.81767
##  [6,] 31.41429 29.30419 31.54471 31.38443 31.39167
```

```
##  [7,] 28.67047 30.71442 27.43962 28.02584 28.02847
##  [8,] 31.15016 33.77543 31.77848 31.99094 31.99034
##  [9,] 31.26934 32.11519 31.75414 31.88843 31.89104
## [10,] 29.71177 29.68247 30.23614 30.20123 30.20554
## [11,] 30.55736 29.48705 31.07133 30.40127 30.40319
## [12,] 31.23148 30.60401 30.96512 31.04992 31.04989
## [13,] 33.49461 32.80955 34.05326 33.68079 33.67567
## [14,] 28.79131 26.68846 30.76405 30.42202 30.39948
## [15,] 37.45842 35.19456 37.31210 36.71131 36.71343
## [16,] 30.46894 28.60251 29.54361 29.78401 29.78255
## [17,] 33.92416 34.24453 33.01843 33.16467 33.16567
## [18,] 33.59733 36.28779 32.70901 32.93896 32.95553
## [19,] 27.20836 25.19622 27.84333 27.56120 27.55121
## [20,] 31.32246 28.37169 31.07154 30.46238 30.46508
## [21,] 26.92613 26.12061 27.50699 27.17705 27.17339
## [22,] 29.71734 31.21595 29.97486 30.27984 30.26991
## [23,] 24.68178 24.45444 25.43191 25.23264 25.22353
## [24,] 29.33845 27.75464 30.43271 29.96762 29.96278
## [25,] 25.06882 21.90242 24.24820 24.05456 24.06572
## [26,] 29.84061 30.87696 29.72589 29.85029 29.85041
## [27,] 30.45037 29.61312 29.70305 29.87953 29.87948
## [28,] 29.22023 28.75997 29.46792 29.53140 29.52896
## [29,] 32.48217 31.72186 32.35945 32.39906 32.40016
## [30,] 27.99053 29.68558 28.25792 28.34775 28.34692
## [31,] 30.56109 37.68943 32.11714 32.36183 32.35235
## [32,] 28.36320 28.07597 28.32501 28.31192 28.31320
## [33,] 28.21122 25.42300 27.44863 28.05411 28.05249
## [34,] 31.65339 30.60883 31.82898 31.04647 31.04937
## [35,] 29.33457 31.24067 31.09465 31.34536 31.32208
## [36,] 26.74064 26.38911 27.03348 27.17884 27.17719
## [37,] 29.69460 29.03112 29.58784 29.63515 29.63458
## [38,] 29.53947 31.56038 28.94898 29.35736 29.34706
## [39,] 28.75444 28.99118 29.17076 28.57937 28.57974
## [40,] 27.34261 26.27024 26.22084 27.02832 27.02985
## [41,] 27.68098 29.40335 27.77731 27.44155 27.44154
## [42,] 32.45601 31.03070 32.99519 33.24809 33.24067
## [43,] 29.87635 30.26872 29.06747 29.38232 29.38320
## [44,] 33.37532 34.40945 33.89061 34.56842 34.56455
## [45,] 27.03778 25.37188 26.36296 26.92623 26.92309
## [46,] 30.74369 35.39858 31.98482 32.46003 32.45413
## [47,] 28.72994 27.56766 28.46341 28.34641 28.34545
## [48,] 30.76437 29.52710 30.42269 30.48293 30.48655
## [49,] 30.56595 30.59279 30.45556 30.35202 30.35320
## [50,] 30.96590 28.76528 30.55719 30.58263 30.59045
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    1    1
##  [2,]    1    1    1    1    1
##  [3,]    1    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    1    1    1    1    1
##  [7,]    1    1    1    1    1
```

```
##  [8,]    1    1    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    1    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    1    1    1    1    1
## [13,]    1    1    1    1    1
## [14,]    1    1    1    1    1
## [15,]    0    0    0    0    0
## [16,]    1    1    1    1    1
## [17,]    1    1    1    1    1
## [18,]    0    0    1    1    1
## [19,]    1    1    1    1    1
## [20,]    1    1    1    1    1
## [21,]    1    1    1    1    1
## [22,]    1    1    1    1    1
## [23,]    0    1    1    0    0
## [24,]    1    1    1    1    1
## [25,]    0    0    0    0    0
## [26,]    1    1    1    1    1
## [27,]    1    1    1    1    1
## [28,]    1    1    1    1    1
## [29,]    1    1    1    1    1
## [30,]    1    1    1    1    1
## [31,]    1    1    1    1    1
## [32,]    1    1    1    1    1
## [33,]    1    1    1    1    1
## [34,]    1    1    1    1    1
## [35,]    1    1    1    1    1
## [36,]    1    1    1    1    1
## [37,]    1    1    1    1    1
## [38,]    1    1    1    1    1
## [39,]    1    1    1    1    1
## [40,]    1    1    1    1    1
## [41,]    1    1    1    1    1
## [42,]    1    1    1    1    1
## [43,]    1    1    0    0    0
## [44,]    1    0    1    1    1
## [45,]    1    1    1    1    1
## [46,]    1    0    1    1    1
## [47,]    1    1    1    1    1
## [48,]    1    1    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    1    1    1    1
```

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

```
##    truth        1        2        3        4        5
## 29.92889 29.88340 29.79287 29.92479 29.94366 29.94302
```

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

```
## [1] -0.045485204 -0.136015125 -0.004095399  0.014768940  0.014134364
```

```
Avg_abs_bias_k0 = colMeans(abs(Bias))
Avg_abs_bias_k0
```

```
## [1] 2.110095 2.777119 2.289013 2.257671 2.258014
```

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

```
## [1] 0.92 0.90 0.94 0.92 0.92
```

```
Avg_res_k0 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0,
                  Avg_bias = Avg_bias_k0, Avg_abs_bias = Avg_abs_bias_k0,
                  Avg_cov = Avg_cov_k0)
```

Then we try to run under different k.

```
#tic()
k <- 0.05

# Matrix for coverage indicators
M <- 50

seed = floor(20*k*M+343)

set.seed(seed)

# Number of estimates
num_estimates <- 5

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,
```

```r
                                check_inclusion, tate_true_m)

  # Store outputs
  true_tates[m] <- tate_true_m
  tate_matrix[m, ] <- tates_m
  coverage_matrix[m, ] <- coverage_indicators
}
#toc()# 357.05 sec elapsed

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

```r
result_k0.05
```

```
## $true_tates
##  [1] 31.04043 27.40843 29.33358 29.84338 30.93604 29.72588 30.07836 30.28437
##  [9] 29.95154 29.45830 27.37609 30.13502 29.05362 28.97050 29.79413 29.29728
## [17] 30.64980 29.34324 29.82895 30.53265 29.91078 28.45548 30.68106 29.73161
## [25] 31.59154 30.20653 30.67365 30.50225 30.58361 32.63924 26.50036 27.55534
## [33] 28.76469 30.69876 31.12003 29.24359 30.58880 28.14092 30.36751 31.26776
## [41] 29.68394 29.59119 26.45765 32.01056 30.12076 31.51440 30.12871 31.13145
## [49] 29.07162 29.76244
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 29.17568 27.27158 28.04125 28.16961 28.18018
##  [2,] 34.22571 33.21494 35.28131 34.85186 34.85815
##  [3,] 28.10066 25.52771 28.09729 28.28075 28.27623
##  [4,] 32.26602 30.81000 32.29393 32.01453 32.01191
##  [5,] 34.58913 35.04493 35.65369 35.60820 35.60185
##  [6,] 29.22861 36.49182 30.12078 30.16522 30.14674
##  [7,] 34.89695 38.27463 36.64933 36.84611 36.84120
##  [8,] 28.50426 24.88720 29.46882 28.90617 28.88586
##  [9,] 30.83395 28.34925 30.28474 30.18475 30.18985
## [10,] 29.44570 29.81163 28.71877 28.98719 28.98657
## [11,] 28.49120 32.12593 29.20155 29.35869 29.33841
## [12,] 26.87266 27.56797 27.64934 27.57012 27.56471
## [13,] 30.28248 34.08933 32.18389 31.94335 31.92560
## [14,] 31.70880 32.79110 31.50415 31.73242 31.73451
## [15,] 34.17476 33.91468 33.51948 33.71593 33.71983
## [16,] 37.00676 36.33293 37.20408 36.91146 36.91278
## [17,] 26.87740 23.42402 26.32163 26.15196 26.16878
## [18,] 30.75047 29.20443 31.88204 31.30283 31.29536
## [19,] 28.37793 32.45336 29.51530 29.65044 29.64776
## [20,] 29.42640 28.35566 30.06663 30.12264 30.12623
## [21,] 33.29382 37.77400 33.08379 33.62640 33.63495
## [22,] 32.36682 33.27689 33.00226 32.72084 32.71744
## [23,] 33.57631 33.67673 34.24600 34.23997 34.23038
## [24,] 33.77620 36.74076 34.29914 34.50101 34.50481
## [25,] 30.79678 32.73568 31.20675 31.30831 31.30633
## [26,] 28.82623 27.96117 27.78541 28.06790 28.06767
```

```
## [27,] 24.37441 25.13028 24.52187 24.51824 24.51785
## [28,] 28.74559 26.75583 28.38931 27.78368 27.80452
## [29,] 28.11952 24.46253 27.11438 28.45423 28.45663
## [30,] 27.69165 32.65835 28.25155 27.84830 27.84805
## [31,] 29.12792 26.92356 28.38089 28.09991 28.09984
## [32,] 26.98093 25.68738 25.62197 25.92210 25.92920
## [33,] 31.59762 32.18493 31.17603 31.22941 31.22840
## [34,] 29.04652 29.24467 29.15862 29.15342 29.15238
## [35,] 31.48852 30.93770 31.37963 31.36697 31.36671
## [36,] 30.81224 30.72575 31.21576 31.13085 31.13232
## [37,] 26.06602 25.22295 26.94049 27.06855 27.06280
## [38,] 29.76655 30.91426 30.83844 30.72309 30.70926
## [39,] 29.59147 29.94770 29.11573 29.31487 29.31697
## [40,] 24.10197 24.84898 24.06280 24.07122 24.07185
## [41,] 29.44524 34.12625 31.50971 31.41704 31.40918
## [42,] 31.46892 31.13759 31.36577 31.37815 31.37934
## [43,] 30.36635 31.42629 31.38616 31.22148 31.21895
## [44,] 28.06373 29.28686 27.02013 27.41734 27.42073
## [45,] 30.58301 30.75460 31.18778 31.11351 31.11168
## [46,] 29.37645 26.72535 30.19861 29.87680 29.87080
## [47,] 31.31862 30.39702 32.16230 31.97845 31.97917
## [48,] 30.89641 31.03481 31.49601 31.46660 31.45983
## [49,] 25.98998 24.80608 25.07468 24.96112 24.97588
## [50,] 32.05663 29.83556 30.75128 30.74487 30.74181
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    1    1
##  [2,]    0    1    0    0    0
##  [3,]    1    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    1    0    1    1    1
##  [7,]    1    0    0    0    0
##  [8,]    1    0    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    1    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    1    1    1    1    1
## [13,]    1    1    1    1    1
## [14,]    1    1    1    1    1
## [15,]    0    1    1    1    1
## [16,]    0    0    0    0    0
## [17,]    1    1    1    1    1
## [18,]    1    1    1    1    1
## [19,]    1    1    1    1    1
## [20,]    1    1    1    1    1
## [21,]    1    0    1    1    1
## [22,]    1    1    0    0    0
## [23,]    1    1    1    1    1
## [24,]    1    0    1    1    1
## [25,]    1    1    1    1    1
## [26,]    1    1    1    1    1
## [27,]    0    1    0    0    0
```

```
## [28,]    1    1    1    1    1
## [29,]    1    0    1    1    1
## [30,]    0    1    1    1    1
## [31,]    1    1    1    1    1
## [32,]    1    1    1    1    1
## [33,]    1    1    1    1    1
## [34,]    1    1    1    1    1
## [35,]    1    1    1    1    1
## [36,]    1    1    1    1    1
## [37,]    1    1    1    1    1
## [38,]    1    1    1    1    1
## [39,]    1    1    1    1    1
## [40,]    0    1    0    0    0
## [41,]    1    1    1    1    1
## [42,]    1    1    1    1    1
## [43,]    1    1    0    1    1
## [44,]    1    1    0    0    0
## [45,]    1    1    1    1    1
## [46,]    1    1    1    1    1
## [47,]    1    1    1    1    1
## [48,]    1    1    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    1    1    1    1
```

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
## 29.83476 30.09896 30.34567 30.31202 30.30398 30.30277
```

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

```
## [1] 0.2642033 0.5109158 0.4772681 0.4692211 0.4680090
```

```
Avg_abs_bias_k0.05 = colMeans(abs(Bias))
Avg_abs_bias_k0.05
```

```
## [1] 2.542352 3.387827 2.805002 2.746409 2.743876
```

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

```
## [1] 0.88 0.86 0.84 0.86 0.86
```

```
Avg_res_k0.05 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.05,
                     Avg_bias = Avg_bias_k0.05, Avg_abs_bias = Avg_abs_bias_k0.05,
                     Avg_cov = Avg_cov_k0.05)
```

I also tried a different k

```
k <- 0.2

# Matrix for coverage indicators
M <- 50

seed = floor(20*k*M+343)

set.seed(seed)

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

```
result_k0.2
```

```
## $true_tates
##  [1] 31.45096 29.28947 32.19479 28.50773 32.93196 30.27356 28.15350 30.84137
##  [9] 29.41652 28.43851 31.64682 31.69475 30.07512 30.61022 29.96724 32.11168
## [17] 28.60928 28.66163 29.59227 28.62718 30.91390 27.34731 32.75405 29.60789
```

```
## [25] 28.11353 30.19661 31.56370 29.81660 30.93565 28.40435 30.45244 27.33632
## [33] 30.97696 30.50510 28.50322 29.99236 30.64045 31.17537 31.81004 33.15640
## [41] 30.94439 29.34637 32.62358 29.09802 31.13591 30.34243 27.59096 30.24423
## [49] 29.64947 30.60226
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 39.86537 41.59893 38.59575 39.03329 39.01659
##  [2,] 21.90341 17.20652 19.16422 19.85324 19.88036
##  [3,] 30.88273 29.78571 30.72250 29.99425 29.99518
##  [4,] 31.17519 27.26393 29.78055 30.48252 30.49432
##  [5,] 34.58930 34.39540 35.02790 35.43974 35.43687
##  [6,] 26.48221 31.09941 26.51118 27.08508 27.09311
##  [7,] 24.86896 29.14640 25.63811 25.27712 25.29639
##  [8,] 30.14571 32.32522 30.87129 30.70882 30.71425
##  [9,] 33.76831 35.61243 32.29431 32.18881 32.21151
## [10,] 30.99889 35.49680 31.75315 31.49814 31.51469
## [11,] 31.97617 32.60102 31.52379 31.39725 31.40764
## [12,] 30.43241 31.88293 29.73137 30.36765 30.38765
## [13,] 31.59419 35.85902 29.67602 29.19543 29.31662
## [14,] 32.55759 32.89214 32.05954 32.05124 32.05542
## [15,] 31.73290 30.33940 31.34270 31.25309 31.24943
## [16,] 27.27392 25.26828 28.43781 28.01108 27.97440
## [17,] 31.65073 32.68128 32.19130 31.90832 31.90742
## [18,] 26.26917 28.59936 25.96509 25.82384 25.82833
## [19,] 29.89753 22.33272 29.21668 29.15231 29.15834
## [20,] 29.58226 31.40486 30.18020 29.90041 29.90141
## [21,] 27.91525 28.89135 28.19187 27.90501 27.90714
## [22,] 26.75848 30.31768 26.63093 26.73371 26.72653
## [23,] 30.61214 30.63804 30.38752 30.58301 30.58324
## [24,] 30.53158 25.89144 31.34676 30.73818 30.74522
## [25,] 26.73066 24.86996 24.60286 24.71449 24.71905
## [26,] 29.27320 29.67165 29.72974 29.77515 29.77149
## [27,] 28.53818 26.54670 26.86940 26.94162 26.96905
## [28,] 25.01288 20.44203 23.60421 23.11473 23.11050
## [29,] 28.84802 29.59733 28.48837 28.67016 28.67736
## [30,] 28.57716 24.83578 26.09920 26.29208 26.32848
## [31,] 36.71029 37.68156 38.13768 37.87696 37.86411
## [32,] 32.68544 30.03628 32.26091 32.66050 32.66065
## [33,] 33.95773 38.29947 35.80232 35.46682 35.47827
## [34,] 26.63804 24.60324 28.65460 28.45189 28.42653
## [35,] 27.86933 28.79864 28.12389 28.01069 28.01097
## [36,] 34.57669 29.67169 34.56932 34.61599 34.59988
## [37,] 27.66280 26.76313 28.00283 27.89410 27.89411
## [38,] 32.42613 32.12415 32.12877 31.97638 31.97427
## [39,] 32.41504 30.10716 33.10191 32.74204 32.73198
## [40,] 29.92318 22.65713 30.35630 30.30772 30.33394
## [41,] 26.39631 31.47845 26.31196 26.81386 26.78337
## [42,] 21.29186 16.55485 21.85017 21.64501 21.61919
## [43,] 29.60979 28.17810 29.96757 29.81839 29.82579
## [44,] 26.13863 27.97406 25.31288 25.35387 25.35635
## [45,] 36.14353 35.80664 36.70075 36.98473 36.97447
## [46,] 31.14286 36.33923 31.44642 32.10123 32.07666
## [47,] 25.86914 27.93889 26.58159 26.56245 26.56454
```

```
## [48,] 27.82173 25.85227 27.34994 26.96966 26.99472
## [49,] 29.71971 29.65253 29.78293 29.54509 29.54610
## [50,] 24.89959 23.71617 25.59299 25.40302 25.41108
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    0    0    1    0    0
##  [2,]    0    0    0    0    0
##  [3,]    1    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    1    1    1    1    1
##  [7,]    1    1    1    1    1
##  [8,]    1    1    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    1    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    1    1    1    1    1
## [13,]    1    1    1    1    1
## [14,]    1    1    1    1    1
## [15,]    1    1    1    1    1
## [16,]    0    0    1    1    1
## [17,]    1    1    1    1    1
## [18,]    1    1    1    1    1
## [19,]    1    1    1    1    1
## [20,]    1    1    1    1    1
## [21,]    1    1    1    1    1
## [22,]    1    1    1    1    1
## [23,]    1    1    1    1    1
## [24,]    1    1    1    1    1
## [25,]    1    1    1    1    1
## [26,]    1    1    1    1    1
## [27,]    1    1    1    1    1
## [28,]    0    0    0    0    0
## [29,]    1    1    1    1    1
## [30,]    1    1    1    1    1
## [31,]    1    1    0    0    0
## [32,]    0    1    0    0    0
## [33,]    1    0    0    1    1
## [34,]    1    1    1    1    1
## [35,]    1    1    1    1    1
## [36,]    0    1    0    0    0
## [37,]    1    1    1    1    1
## [38,]    1    1    1    1    1
## [39,]    1    1    1    1    1
## [40,]    1    0    1    1    1
## [41,]    1    1    0    1    1
## [42,]    0    0    0    0    0
## [43,]    1    1    1    1    1
## [44,]    1    1    1    1    1
## [45,]    0    1    1    1    1
## [46,]    1    1    1    1    1
## [47,]    1    1    1    1    1
## [48,]    1    1    1    1    1
```

```
## [49,]    1    1    1    1    1
## [50,]    1    1    1    1    1
```

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
## 30.17749 29.68685 29.47455 29.57340 29.54580 29.55010
```

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

```
## [1] -0.4906425 -0.7029417 -0.6040882 -0.6316855 -0.6273893
```

```
Avg_abs_bias_k0.2 = colMeans(abs(Bias))
Avg_abs_bias_k0.2
```

```
## [1] 2.791333 3.807623 2.903860 2.932849 2.927544
```

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

```
## [1] 0.84 0.86 0.84 0.86 0.86
```

```
Avg_res_k0.2 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.2,
                    Avg_bias = Avg_bias_k0.2, Avg_abs_bias = Avg_abs_bias_k0.2,
                    Avg_cov = Avg_cov_k0.2)
```

```
k <- 0.5

# Matrix for coverage indicators
M <- 50

seed = floor(20*k*M+343)

set.seed(seed)

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)
```

```r
  # 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_k0.5 <- list(true_tates, tate_matrix, coverage_matrix)
names(result_k0.5) <- c("true_tates", "tate_matrix", "coverage_matrix")
```

Display result at k=0.5

```r
result_k0.5
```

```
## $true_tates
##  [1] 28.16377 29.39766 30.16217 30.32831 29.96225 30.83302 33.14148 29.87199
##  [9] 30.83922 27.98904 30.70191 27.97405 29.51346 27.44850 30.23894 30.89106
## [17] 29.52916 28.73820 30.34517 27.70388 31.82781 28.19414 30.40061 28.08573
## [25] 30.60007 29.87151 31.96670 28.54538 29.47380 28.91783 28.77737 27.63617
## [33] 30.34377 29.90514 29.99256 29.05825 29.15715 30.52666 30.50064 29.04773
## [41] 30.25072 30.72473 29.15331 29.54090 29.87894 30.94926 30.76848 28.86120
## [49] 30.29142 31.19572
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 27.37421 22.52885 26.63523 26.90478 26.94729
##  [2,] 35.95854 34.62113 32.38759 32.03108 32.11107
##  [3,] 21.24570 14.82241 23.39324 22.26860 22.28439
##  [4,] 28.67587 25.79268 28.58498 27.53678 27.58344
##  [5,] 33.11749 27.34569 34.99702 34.85653 34.76244
##  [6,] 36.92102 22.65440 35.82264 36.02853 35.80517
##  [7,] 31.02375 18.97836 28.92332 30.61214 30.62952
##  [8,] 30.40225 30.16659 35.92656 35.20387 35.09102
##  [9,] 28.90209 21.17279 28.46370 27.93996 27.91409
## [10,] 20.86934 18.74493 20.47472 19.88615 19.88199
## [11,] 32.92497 37.83444 33.88682 34.26127 34.28862
## [12,] 24.30629 11.19601 20.26321 20.14793 20.13808
## [13,] 24.91218 28.10106 21.34620 21.24467 21.26186
## [14,] 32.32529 25.99590 27.53985 27.93077 27.83040
## [15,] 27.04850 28.64035 25.88632 26.70695 26.72852
## [16,] 34.32032 30.44214 33.39108 33.17516 33.12364
## [17,] 29.98857 44.77209 26.51667 26.81817 26.77177
## [18,] 31.98665 37.75969 32.36133 30.81020 30.83647
```

```
## [19,] 32.00405 18.58964 32.99587 31.55087 31.60401
## [20,] 23.61208 24.79156 24.59203 25.13875 25.13082
## [21,] 23.70644 22.55524 23.19359 23.44809 23.44097
## [22,] 27.21818 17.47281 24.13985 22.10186 22.24098
## [23,] 23.28743 22.25852 24.33780 23.56315 23.55452
## [24,] 31.95352 31.67117 30.06830 30.79553 30.77862
## [25,] 35.34344 29.17893 34.58864 33.76382 33.81159
## [26,] 20.96570 21.67280 18.51715 17.73553 17.74601
## [27,] 26.18139 26.17782 24.48098 24.98111 24.98755
## [28,] 32.72549 38.53695 30.75224 30.47302 30.45694
## [29,] 23.45081 21.93344 22.25468 23.03786 23.03788
## [30,] 23.45589 37.83072 22.57417 23.79836 23.77819
## [31,] 35.25033 29.08915 35.81295 34.92882 35.00868
## [32,] 43.34030 44.72978 42.68307 43.24609 43.25192
## [33,] 26.99271 24.11243 24.27961 24.13407 24.20642
## [34,] 34.94707 32.47609 32.39643 33.00762 33.01044
## [35,] 34.28087 32.59663 33.70011 34.45388 34.45833
## [36,] 30.14121 29.57856 32.39378 31.11651 31.12238
## [37,] 27.20604 24.49976 26.01252 25.56075 25.53969
## [38,] 30.26904 35.40817 27.33834 28.71776 28.73297
## [39,] 23.51740 28.01268 23.52632 23.22799 23.21645
## [40,] 24.29093 22.63614 25.16803 25.29318 25.32474
## [41,] 32.14682 27.14830 31.59215 32.35515 32.35619
## [42,] 25.60041 29.76396 28.97039 28.62428 28.63917
## [43,] 27.53230 22.28157 24.33480 25.67478 25.70948
## [44,] 26.36658 26.11821 26.46665 25.99052 25.99767
## [45,] 32.23732 42.02787 30.65553 31.12729 31.01720
## [46,] 33.11089 35.50093 34.48587 34.74312 34.74406
## [47,] 29.83949 24.50969 34.99626 32.08146 31.94916
## [48,] 28.00683 38.51347 27.51394 28.23650 28.28957
## [49,] 22.45832 22.65515 23.71503 22.83125 22.84504
## [50,] 29.97019 32.71578 28.36210 28.49604 28.48099
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    1    1    1
##  [2,]    1    1    1    1    1
##  [3,]    0    0    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    1    1    1    1    1
##  [7,]    1    1    1    1    1
##  [8,]    1    1    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    0    1    0    0    0
## [11,]    1    1    1    1    1
## [12,]    1    0    0    0    0
## [13,]    1    1    0    0    0
## [14,]    1    1    1    1    1
## [15,]    1    1    1    1    1
## [16,]    1    1    1    1    1
## [17,]    1    1    1    1    1
## [18,]    1    1    1    1    1
## [19,]    1    1    1    1    1
```

```
## [20,]    1    1    1    1    1
## [21,]    0    0    0    0    0
## [22,]    1    1    1    1    1
## [23,]    0    1    1    1    1
## [24,]    1    1    1    1    1
## [25,]    1    1    1    1    1
## [26,]    0    1    0    0    0
## [27,]    1    1    1    1    1
## [28,]    1    1    1    1    1
## [29,]    0    1    0    1    1
## [30,]    1    1    1    1    1
## [31,]    0    1    0    0    0
## [32,]    0    0    0    0    0
## [33,]    1    1    1    0    0
## [34,]    0    1    1    1    1
## [35,]    1    1    1    1    1
## [36,]    1    1    1    1    1
## [37,]    1    1    1    1    1
## [38,]    1    1    1    1    1
## [39,]    0    1    1    1    1
## [40,]    1    1    1    1    1
## [41,]    1    1    1    1    1
## [42,]    1    1    1    1    1
## [43,]    1    1    1    1    1
## [44,]    1    1    1    1    1
## [45,]    1    0    1    1    1
## [46,]    1    1    1    1    1
## [47,]    1    1    1    1    1
## [48,]    1    1    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    1    1    1    1
```

```
Truth_W_tate = cbind.data.frame(truth = as.matrix(result_k0.5$true_tates,nrow=M),result_k0.5$tate_matri
Avg_truth_W_tate_k0.5 = colMeans(Truth_W_tate)
Avg_truth_W_tate_k0.5
```

```
##    truth        1        2        3        4        5
## 29.76434 29.07425 27.97227 28.47399 28.37197 28.36917
```

```
Bias = result_k0.5$tate_matrix - result_k0.5$true_tates
Avg_bias_k0.5 = colMeans(Bias)
Avg_bias_k0.5
```

```
## [1] -0.690089 -1.792071 -1.290346 -1.392368 -1.395170
```

```
Avg_abs_bias_k0.5 = colMeans(abs(Bias))
Avg_abs_bias_k0.5
```

```
## [1] 4.051205 6.402473 4.522196 4.404479 4.386392
```

```r
Avg_cov_k0.5 = colMeans(result_k0.5$coverage_matrix)
Avg_cov_k0.5
```

```
## [1] 0.80 0.90 0.84 0.84 0.84
```

```r
Avg_res_k0.5 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.5,
                    Avg_bias = Avg_bias_k0.5, Avg_abs_bias = Avg_abs_bias_k0.5,
                    Avg_cov = Avg_cov_k0.5)
```

```r
k <- 0.8

# Matrix for coverage indicators
M <- 50

seed = floor(20*k*M+343)

set.seed(seed)

# Number of estimates
num_estimates <- 5

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] 28.34039 26.95064 31.02354 29.50911 31.10201 27.65848 29.30633 30.72133
##  [9] 27.95384 30.04981 31.37843 29.60775 30.74731 31.52304 29.47856 30.38838
## [17] 29.50304 31.08556 29.26279 31.52882 28.97565 27.62442 30.29708 29.42332
## [25] 30.36889 31.77990 27.33848 30.07191 28.93702 29.26992 32.55207 30.46321
## [33] 29.33423 30.54502 28.29815 26.27319 30.56849 30.16205 30.78964 31.86781
## [41] 28.99664 31.78434 30.38072 29.52692 29.55730 31.92481 28.12242 31.06683
## [49] 27.41024 28.26102
##
## $tate_matrix
##            [,1]       [,2]      [,3]      [,4]      [,5]
##  [1,] 45.173387 37.6651228 39.750616 38.804430 38.714032
##  [2,] 35.631563 17.2030742 39.228421 36.472429 36.704202
##  [3,] 20.570763 36.5785544 27.735196 27.552561 27.152073
##  [4,] 20.070906  7.5170409 15.636000 15.199152 15.297752
##  [5,] 32.598516 47.8713999 32.846230 32.103379 31.795339
##  [6,] 35.874466 33.1021140 38.049977 39.013914 39.203336
##  [7,] 20.741450 30.0991708 27.338521 26.749786 26.706081
##  [8,] 12.241311 -0.7067555 22.528244 20.692876 20.311008
##  [9,] 28.327355 49.4686993 28.919504 27.594205 27.624474
## [10,] 29.045875 35.0403402 26.614464 28.622875 28.503180
## [11,] 28.539005 35.5837687 29.551436 28.529808 28.491621
## [12,] 36.457142 60.2806237 40.403848 43.777501 43.464751
## [13,] 15.688095 19.2128629 20.443902 19.451316 19.529467
## [14,] 26.241476 44.5874661 39.155470 39.085723 38.972317
## [15,] 41.217899 16.7871994 35.481591 34.886250 34.386001
## [16,] 31.599198 36.7909261 32.725982 33.434470 33.451389
## [17,] 13.615512 31.1429424 14.203119 15.473654 15.940803
## [18,] 45.514377 45.3279648 39.289708 40.008209 39.990616
## [19,] 15.375028 -0.4437831 12.636215 11.787233 11.963515
## [20,] 37.223071 34.5976521 37.879612 36.436723 36.408813
## [21,] 46.163979 52.6285558 51.132125 51.235290 50.969307
## [22,] 31.506155 38.1526037 32.844290 32.131791 32.278102
## [23,] 19.988727 18.7919037 16.046158 18.226581 18.178265
## [24,] 37.927896 60.8955839 34.960057 32.986426 33.786220
## [25,] 17.896441  1.7667461 18.827582 19.904493 20.082194
## [26,] 29.733108 23.8603034 33.342423 32.833719 32.872879
## [27,] 31.178674 14.9502858 27.840432 26.468551 26.454441
## [28,] 20.114512 21.2665542 25.989190 25.910470 25.933765
## [29,] 33.144098 33.4214761 25.069734 23.943657 24.009579
## [30,] 41.457654 20.8102967 40.662687 40.121391 40.232518
## [31,] 38.528968 41.3011222 39.451175 39.674592 39.711949
## [32,] 37.361492 25.7914649 38.044667 36.974731 36.728774
## [33,] 29.811925 40.3512789 33.567373 34.390366 34.266387
## [34,] 25.337412 30.4754439 25.203493 25.899226 25.945217
## [35,] 26.548248 23.0992956 24.939338 23.846781 23.804770
## [36,] 40.177767 32.4965079 47.436579 46.444159 46.479424
## [37,] 44.316831 51.4678528 49.707989 50.187358 50.268841
## [38,]  9.110417  1.2837126  8.427729  8.660089  8.553735
## [39,] 26.835349 46.6380932 26.288346 28.212858 27.833290
## [40,] 23.702630 36.5181731 30.313937 28.935390 29.087817
## [41,] 14.037974 13.1153670 14.583301 15.948093 15.842310
## [42,] 46.541623 42.3652901 43.862130 46.009099 45.983874
## [43,] 25.905270 52.1437092 32.732138 33.647391 33.204084
## [44,] 30.961393 43.8709097 38.741932 38.130734 38.104442
```

```
## [45,] 28.002789 40.9432504 28.406600 28.677317 28.623295
## [46,] 40.515460 43.4089072 46.488835 45.556836 45.534591
## [47,] 28.656388 14.7427132 33.871507 34.527215 34.386791
## [48,] 38.536907 31.3211022 35.624626 35.596569 35.546742
## [49,] 25.915357 17.5161834 25.036880 25.613894 25.551349
## [50,] 26.197208 57.8531218 36.429516 36.713537 37.292242
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    0    1    1    1    1
##  [2,]    1    1    1    1    1
##  [3,]    0    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    1    1    1    0    0
##  [7,]    1    1    1    1    1
##  [8,]    0    0    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    1    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    1    1    0    0    0
## [13,]    0    1    1    0    0
## [14,]    1    1    1    1    1
## [15,]    1    1    1    1    1
## [16,]    1    1    1    1    1
## [17,]    0    1    0    1    1
## [18,]    0    1    1    1    1
## [19,]    1    1    1    1    1
## [20,]    1    1    1    1    1
## [21,]    0    0    0    0    0
## [22,]    1    1    1    1    1
## [23,]    1    1    0    1    1
## [24,]    1    1    1    1    1
## [25,]    0    0    0    1    1
## [26,]    1    1    1    1    1
## [27,]    1    1    1    1    1
## [28,]    1    1    1    1    1
## [29,]    1    1    1    1    1
## [30,]    0    1    1    1    1
## [31,]    1    1    1    1    1
## [32,]    1    1    1    1    1
## [33,]    1    1    1    1    1
## [34,]    1    1    1    1    1
## [35,]    1    1    1    1    1
## [36,]    0    1    0    0    0
## [37,]    0    1    0    0    0
## [38,]    0    0    0    0    0
## [39,]    1    1    1    1    1
## [40,]    1    1    1    1    1
## [41,]    0    1    0    0    0
## [42,]    0    1    0    0    0
## [43,]    1    1    1    1    1
## [44,]    1    1    1    1    1
## [45,]    1    1    1    1    1
```

```
## [46,]    1    1    0    0    0
## [47,]    1    1    1    1    1
## [48,]    1    1    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    0    1    1    1
```

```
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
## 29.78182 29.75718 31.81908 31.32582 31.26170 31.24316
```

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

```
## [1] -0.02463581  2.03726714  1.54399972  1.47988523  1.46134263
```

```
Avg_abs_bias_k0.8 = colMeans(abs(Bias))
Avg_abs_bias_k0.8
```

```
## [1]  7.989614 12.947416  8.063362  7.967681  7.968373
```

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

```
## [1] 0.72 0.90 0.78 0.80 0.80
```

```
Avg_res_k0.8 = list(Avg_truth_W_tate = Avg_truth_W_tate_k0.8,
                    Avg_bias = Avg_bias_k0.8, Avg_abs_bias = Avg_abs_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$Avg_abs_bias, avg_res$Avg_cov), ncol=4,nrow = 5)
vec = round(c(as.numeric(avg_res$Avg_truth_W_tate[1]),c(t(dt))),4)
return(vec)
}

avg_vec_k0 = res2vec(Avg_res_k0)
avg_vec_k0.05 = res2vec(Avg_res_k0.05)
avg_vec_k0.2 = res2vec(Avg_res_k0.2)
avg_vec_k0.5 = res2vec(Avg_res_k0.5)
avg_vec_k0.8 = res2vec(Avg_res_k0.8)
```

```r
avg_tb = rbind.data.frame(avg_vec_k0,avg_vec_k0.05,avg_vec_k0.2,avg_vec_k0.5,avg_vec_k0.8)

colnames(avg_tb) = c("Truth","ATE_OR","bias_OR","Abs_bias_OR","cov_OR","ATE_IPW",
                     "bias_IPW","Abs_bias_IPW","cov_IPW","ATE_IPW2","bias_IPW2",
                     "Abs_bias_IPW2","cov_IPW2","ATE_DRE1","bias_DRE1",
                     "Abs_bias_DRE1","cov_DRE1","ATE_DRE2","bias_DRE2",
                     "Abs_bias_DRE2","cov_DRE2")

rownames(avg_tb) = c("K=0","K=0.05","K=0.5","K=0.2","K=0.8")

avg_tb
```

```
##          Truth  ATE_OR bias_OR Abs_bias_OR cov_OR ATE_IPW bias_IPW Abs_bias_IPW
## K=0    29.9289 29.8834 -0.0455      2.1101   0.92 29.7929  -0.1360       2.7771
## K=0.05 29.8348 30.0990  0.2642      2.5424   0.88 30.3457   0.5109       3.3878
## K=0.5  30.1775 29.6868 -0.4906      2.7913   0.84 29.4745  -0.7029       3.8076
## K=0.2  29.7643 29.0743 -0.6901      4.0512   0.80 27.9723  -1.7921       6.4025
## K=0.8  29.7818 29.7572 -0.0246      7.9896   0.72 31.8191   2.0373      12.9474
##        cov_IPW ATE_IPW2 bias_IPW2 Abs_bias_IPW2 cov_IPW2 ATE_DRE1 bias_DRE1
## K=0       0.90  29.9248   -0.0041        2.2890     0.94  29.9437    0.0148
## K=0.05    0.86  30.3120    0.4773        2.8050     0.84  30.3040    0.4692
## K=0.5     0.86  29.5734   -0.6041        2.9039     0.84  29.5458   -0.6317
## K=0.2     0.90  28.4740   -1.2903        4.5222     0.84  28.3720   -1.3924
## K=0.8     0.90  31.3258    1.5440        8.0634     0.78  31.2617    1.4799
##        Abs_bias_DRE1 cov_DRE1 ATE_DRE2 bias_DRE2 Abs_bias_DRE2 cov_DRE2
## K=0           2.2577     0.92  29.9430    0.0141        2.2580     0.92
## K=0.05        2.7464     0.86  30.3028    0.4680        2.7439     0.86
## K=0.5         2.9328     0.86  29.5501   -0.6274        2.9275     0.86
## K=0.2         4.4045     0.84  28.3692   -1.3952        4.3864     0.84
## K=0.8         7.9677     0.80  31.2432    1.4613        7.9684     0.80
```

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

Separate 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    29.9289 29.8834 29.7929  29.9248  29.9437  29.9430
## K=0.05 29.8348 30.0990 30.3457  30.3120  30.3040  30.3028
## K=0.5  30.1775 29.6868 29.4745  29.5734  29.5458  29.5501
## K=0.2  29.7643 29.0743 27.9723  28.4740  28.3720  28.3692
## K=0.8  29.7818 29.7572 31.8191  31.3258  31.2617  31.2432
```

```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    -0.0455  -0.1360  -0.0041   0.0148   0.0141
## K=0.05  0.2642   0.5109   0.4773   0.4692   0.4680
## K=0.5  -0.4906  -0.7029  -0.6041  -0.6317  -0.6274
## K=0.2  -0.6901  -1.7921  -1.2903  -1.3924  -1.3952
## K=0.8  -0.0246   2.0373   1.5440   1.4799   1.4613
```

```
Abs_bias_sub = avg_tb %>% select(Abs_bias_OR,Abs_bias_IPW,Abs_bias_IPW2,
                                 Abs_bias_DRE1,Abs_bias_DRE2)
Abs_bias_sub
```

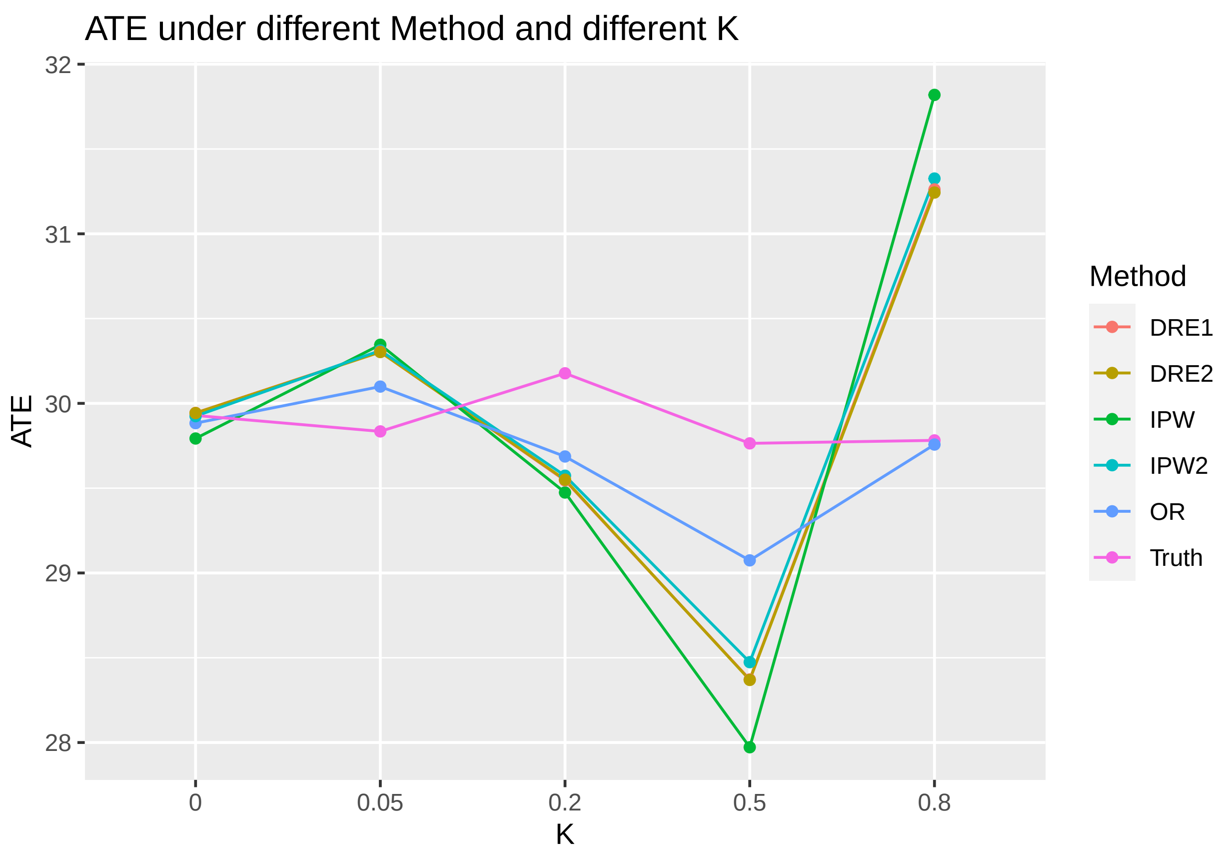
```
##        Abs_bias_OR Abs_bias_IPW Abs_bias_IPW2 Abs_bias_DRE1 Abs_bias_DRE2
## K=0         2.1101       2.7771        2.2890        2.2577        2.2580
## K=0.05      2.5424       3.3878        2.8050        2.7464        2.7439
## K=0.5       2.7913       3.8076        2.9039        2.9328        2.9275
## K=0.2       4.0512       6.4025        4.5222        4.4045        4.3864
## K=0.8       7.9896      12.9474        8.0634        7.9677        7.9684
```

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

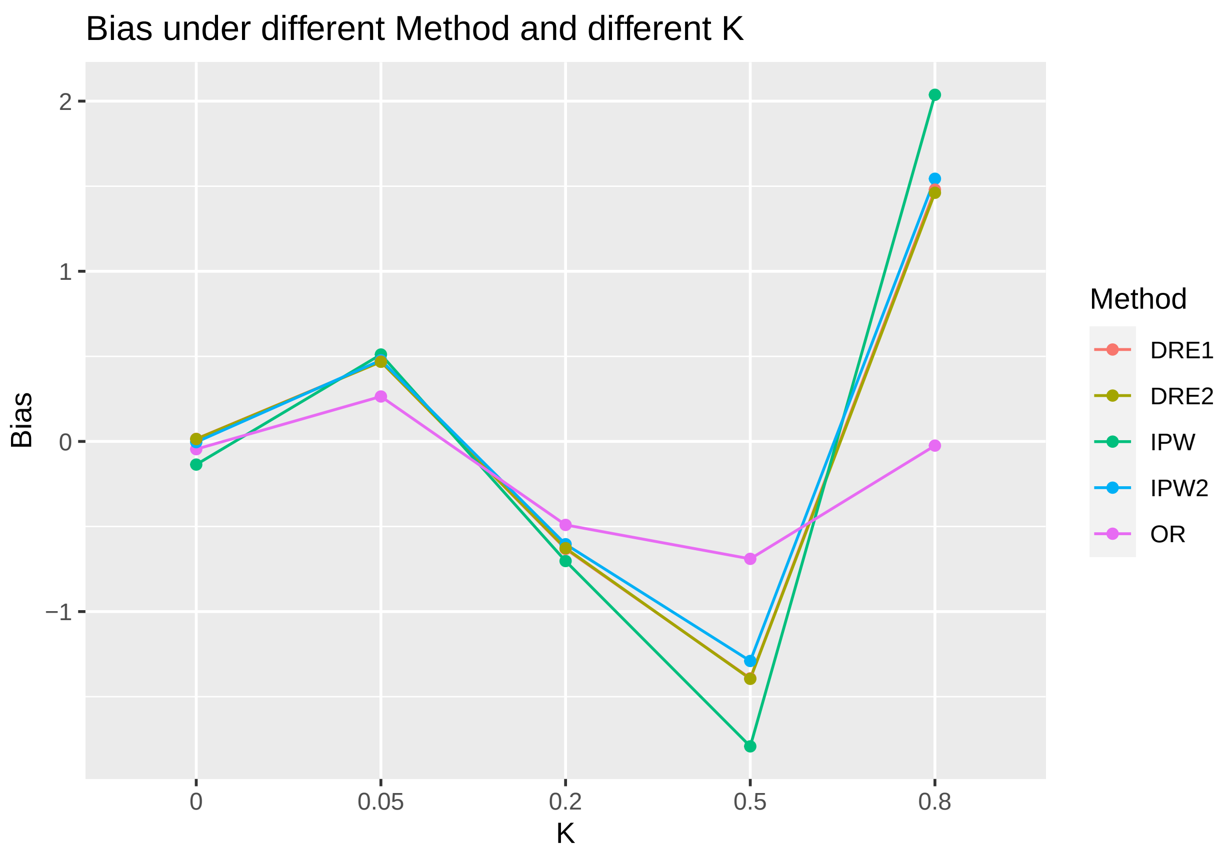
```
##        cov_OR cov_IPW cov_IPW2 cov_DRE1 cov_DRE2
## K=0      0.92    0.90     0.94     0.92     0.92
## K=0.05   0.88    0.86     0.84     0.86     0.86
## K=0.5    0.84    0.86     0.84     0.86     0.86
## K=0.2    0.80    0.90     0.84     0.84     0.84
## K=0.8    0.72    0.90     0.78     0.80     0.80
```

Here is for graphing

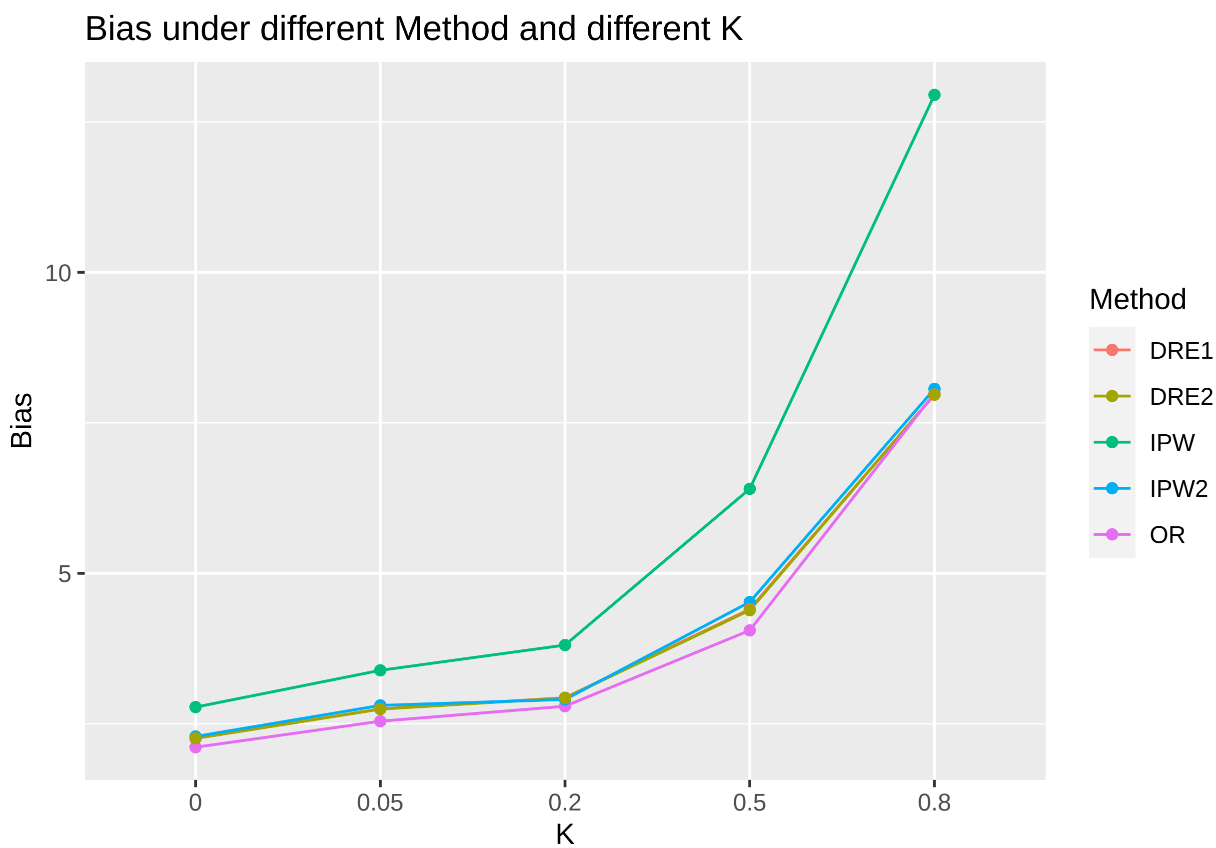
```
k_lst = c("0","0.05","0.2","0.5","0.8")

##### 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")
```



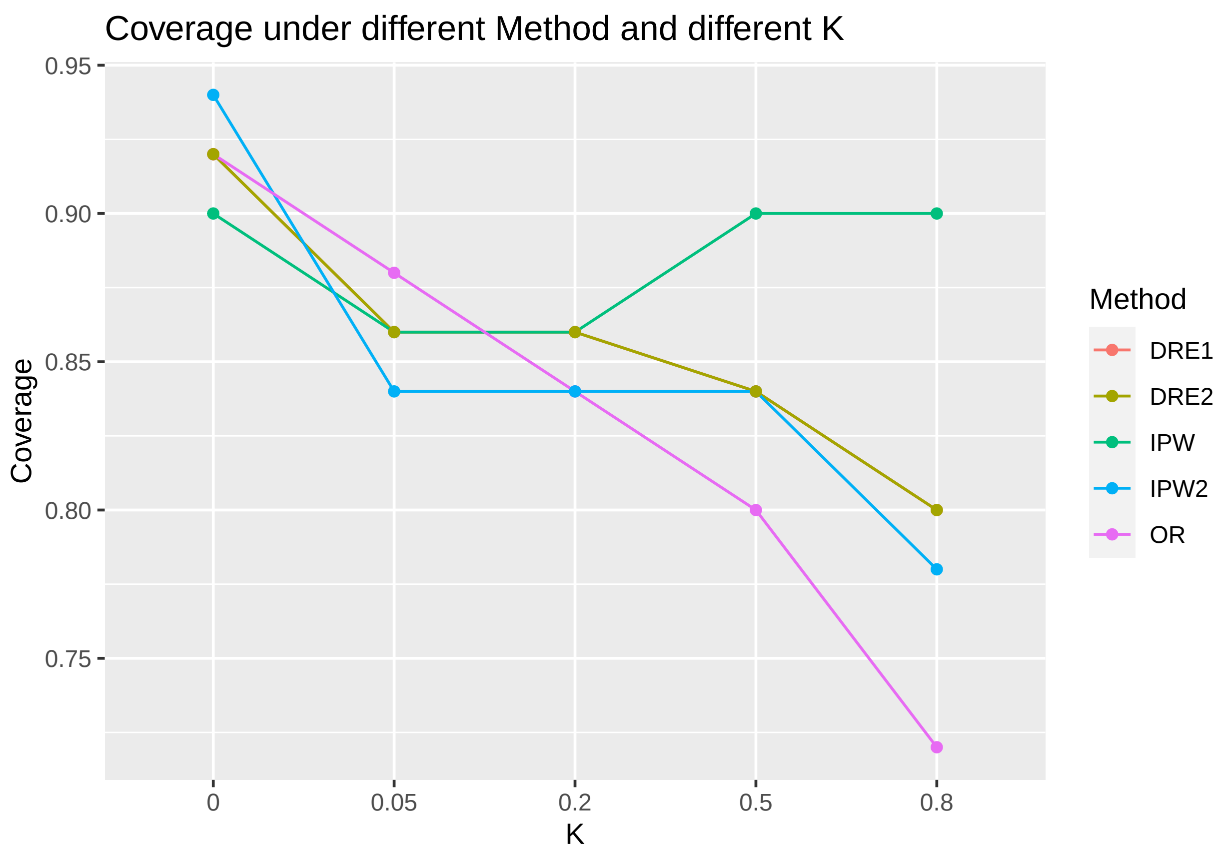

```r
##### Bias #####
method = rep(c("OR" ,"IPW" ,"IPW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,5)
bias_lst = as.numeric(unlist(bias_sub))
bias_tb = cbind.data.frame(Method = method,K = k,bias = bias_lst)

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")
```




```r
##### Absolute Bias #####
method = rep(c("OR" ,"IPW" ,"IPW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,5)
Abs_bias_lst = as.numeric(unlist(Abs_bias_sub))
Abs_bias_tb = cbind.data.frame(Method = method,K = k,Abs_bias = Abs_bias_lst)

library(ggplot2)
ggplot(data=Abs_bias_tb, aes(x=K, y=Abs_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")
```




```r
##### 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
0.95
0.90
0.85
0.80
0.75
Coverage
0
0.05
0.2
0.5
0.8
K
Method
DRE1
DRE2
IPW
IPW2
OR