Casual Project Test Run with optimal k

Tiankai

2023-12-5

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)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2

## Warning: package 'purrr' was built under R version 4.3.2

## Warning: package 'stringr' was built under R version 4.3.2

## Warning: package 'lubridate' was built under R version 4.3.2

## -- 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   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- 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

# function for generate optimal k
optiaml_a1a2 = function(k,mu_trial=mu_trial,mu_target=mu_target){
  #flip = (-1)^rbinom(1,1,prob = 0.5)
  a <- 0.5 - k/2
  b <- 0.5 + k/2

  f =  function (a1) qnorm(a1,mu_trial,1) - qnorm(2*a-a1,mu_target,1)
  optimal_a1 = uniroot(f, lower = 0, upper = min(1,2*a))$root
  a2 = a-optimal_a1
```

```r
  return(list(k=k, a1=optimal_a1, a2=a2))
}

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
```

```r
# 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
Udiff <- runif(1, -1, 1) # Size of underlying var difference between trial and target pops
mu_trial = -Udiff*0.5
mu_target = Udiff*0.5
UTrial <- rnorm(N_Trial, mu_trial, 1)
UTarget <- rnorm(N_Target, mu_target, 1)
U = c(UTrial, UTarget)

a1 = optiaml_a1a2(k=k,mu_trial=mu_trial,mu_target=mu_target)$a1
flip = (-1)^rbinom(1,1,prob = 0.5)
a_thresh <- flip*qnorm(a1,mu_trial,1)
b_thresh <- -a_thresh
VTrial <- ifelse(UTrial < a_thresh, "Low", "High")
VTarget <- ifelse(UTarget < b_thresh, "Low", "High")

# # Cut Latent Variable
# Trial_cut_less <- rbinom(1, 1, prob = 0.5) # Binary indicator for Trial cut point less than Target
# a <- l - k/2 # l=0.5
# 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_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_U <- 3

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

  Y0 <- beta_male * Male + beta_age * Age + beta_bmi * BMI + beta_U * U +
    effect_race + rnorm(N, beta_0, sd = epsilon)
  Y1 <- beta_male * Male + beta_age * Age + beta_bmi * BMI + beta_U * U +
    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, U)

  ## -- 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

  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)
}
```

```
library(tictoc)
```

```
## Warning: package 'tictoc' was built under R version 4.3.1
```

ANALYSIS

Here is the base case.

```
k <- 0

tic()
# 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 <- confidence_interval_length_matrix <- matrix(NA, nrow = M, ncol = num_

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)
  # Get Confidence Interval Length
  confidence_interval_lengths <- confidence_intervals[2, ] - confidence_intervals[1, ]

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

result_k0 <- list(true_tates, tate_matrix, coverage_matrix, confidence_interval_length_matrix)
names(result_k0) <- c("true_tates", "tate_matrix", "coverage_matrix", "confidence_interval_length_matri
toc() #244.52 sec elapsed
```

```
## 349.33 sec elapsed
```

Display result at k=0

```
result_k0
```

```
## $true_tates
##  [1] 33.00623 29.99105 30.12502 30.24467 30.88774 30.85519 30.85987 27.82039
##  [9] 32.04332 29.54960 30.83454 30.84454 34.85854 30.67569 30.18824 31.78383
## [17] 29.93864 29.29309 30.46552 29.69344 31.96415 33.75278 30.55784 27.28204
## [25] 28.79625 25.78506 32.05358 32.49273 28.51430 32.87202 30.41258 28.87308
## [33] 30.39153 30.83513 28.91246 28.76150 35.83301 30.39006 32.63948 27.02021
## [41] 27.46495 30.44102 34.66753 27.76159 31.47505 29.74357 26.36008 29.82470
## [49] 27.26372 27.08257
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 33.21862 42.21221 35.60421 36.18801 36.20270
##  [2,] 30.33273 28.58662 30.96456 30.79926 30.78490
##  [3,] 26.25403 25.14426 26.08633 25.71227 25.71121
##  [4,] 28.76878 28.47456 28.45617 28.45910 28.46273
##  [5,] 27.25900 28.84223 27.12343 27.36237 27.35944
##  [6,] 29.98550 28.16357 29.54105 30.01481 30.00839
##  [7,] 27.53877 26.08979 28.91958 29.04384 29.03186
##  [8,] 27.03679 30.72600 25.75057 26.70954 26.71494
##  [9,] 27.67864 30.00273 26.89172 26.82760 26.83781
## [10,] 31.26284 29.02323 27.52547 28.19998 28.20150
## [11,] 29.79915 29.77393 28.74471 28.93582 28.94337
## [12,] 26.21651 29.33542 26.46061 26.42704 26.42380
## [13,] 33.56716 31.91847 33.52779 33.62411 33.62715
## [14,] 27.14309 28.47470 27.52543 27.67416 27.67143
```

```
## [15,] 33.99905 30.80413 33.12020 33.25419 33.27126
## [16,] 29.97122 30.45209 28.43121 28.74485 28.75227
## [17,] 29.13402 27.40480 28.59211 28.47008 28.47524
## [18,] 31.31706 29.05557 29.82499 29.47136 29.47561
## [19,] 27.08327 27.11719 27.39032 27.35345 27.34755
## [20,] 28.41313 33.53297 28.38599 29.10989 29.09330
## [21,] 28.84777 32.15482 29.61503 29.33629 29.34051
## [22,] 30.23515 28.36264 31.74276 31.62406 31.61687
## [23,] 28.26095 32.44542 30.24692 30.19558 30.16723
## [24,] 29.22632 25.84541 29.02739 28.53682 28.58569
## [25,] 28.05780 25.34614 28.94199 28.43461 28.43440
## [26,] 27.89957 23.87871 28.33874 28.21085 28.21289
## [27,] 28.43199 30.82547 27.38664 27.99268 28.00506
## [28,] 26.32414 27.94932 27.68777 27.67902 27.65221
## [29,] 29.18496 29.06715 30.11803 29.91597 29.90424
## [30,] 30.77117 33.36781 31.59021 31.22905 31.22004
## [31,] 30.51630 28.74926 31.51728 31.05179 31.03816
## [32,] 30.26840 30.10380 30.13535 29.94674 29.94703
## [33,] 27.88390 19.07697 27.98749 28.61719 28.58691
## [34,] 30.86993 25.51463 28.73445 29.14509 29.16866
## [35,] 28.99710 25.37098 28.40303 28.02775 28.02825
## [36,] 26.22646 28.14422 26.36176 26.53035 26.52142
## [37,] 32.88507 26.28777 31.27396 32.90820 32.90610
## [38,] 31.92104 35.42544 32.38457 31.89837 31.92278
## [39,] 37.74146 35.44488 38.35232 37.79223 37.79856
## [40,] 30.12148 32.86932 29.43310 29.60509 29.61684
## [41,] 27.24885 24.06408 27.86736 26.90898 26.91149
## [42,] 30.63989 35.01875 30.23754 30.46914 30.46313
## [43,] 27.98292 30.22980 29.52780 29.65383 29.64286
## [44,] 28.98813 31.48617 30.62759 30.35823 30.33867
## [45,] 32.32816 32.28258 32.80054 33.49572 33.50143
## [46,] 27.79948 30.07855 27.42784 27.96256 27.97884
## [47,] 28.45889 26.51096 28.22986 28.02258 28.02327
## [48,] 32.43505 35.10439 32.73684 32.71302 32.71981
## [49,] 28.53087 26.17274 30.72973 30.47782 30.43603
## [50,] 30.10919 28.52561 33.22762 32.99407 33.00858
##
## $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    0    0    0
## [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,]    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,]    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    1    0    0    0
## [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    0    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,]    0    1    1    1    1
## [44,]    1    1    1    1    1
## [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    0    0    0
##
## $confidence_interval_length_matrix
##            [,1]     [,2]      [,3]      [,4]      [,5]
##  [1,] 11.880200 20.65846 12.841347 12.902212 12.939917
##  [2,]  9.690840 14.58272 11.214152 10.460380 10.428307
##  [3,]  9.936270 12.53525 10.529051 10.001841 10.007254
##  [4,] 11.313263 14.78662 13.017563 12.605106 12.610689
##  [5,] 10.020415 14.34125 10.332061 10.350925 10.349391
##  [6,] 11.239983 13.93091 12.480117 12.242801 12.140965
##  [7,] 10.749867 11.47474 11.304151 10.936522 10.902425
##  [8,] 10.164156 15.25459 11.153018 11.574614 11.563609
##  [9,]  9.296626 11.13782  9.700077  8.959869  8.973679
## [10,] 13.115333 19.02536 14.156485 14.323490 14.296197
## [11,]  8.851848 12.18597  9.289134  8.984639  8.985454
## [12,]  8.594302 13.23175  9.477854  9.432792  9.442943
## [13,] 11.236104 20.79513 10.058930 10.831882 10.787438
## [14,] 10.287436 14.39544 10.803474 10.376412 10.368842
## [15,]  9.596509 14.68446 10.209588 10.546413 10.536565
## [16,]  8.579881 11.50004  8.927905  8.729496  8.703870
```

```
## [17,]  8.814091 11.26979  8.791171  8.626833  8.618004
## [18,] 10.651255 14.96120 10.870944 11.334182 11.308140
## [19,]  7.968422 15.14481  8.244818  7.345712  7.337474
## [20,]  9.002616 12.22747  8.674744  8.947391  8.939097
## [21,] 10.389205 11.95312 11.092727 10.942079 10.935105
## [22,]  8.581146 13.94109  9.652834  9.746518  9.735633
## [23,]  9.158762 14.78926  9.440953  8.994791  8.954876
## [24,] 10.197264 18.20538 11.392213 12.048989 11.857307
## [25,] 11.571801 19.69957 11.402580 11.535147 11.521069
## [26,] 10.454872 12.84143 11.416014 11.451898 11.442282
## [27,]  9.678653 11.34764 10.330152 10.572514 10.552007
## [28,] 10.812302 21.34357 10.607486 10.756286 10.746647
## [29,] 10.108016 11.82056  9.740813  9.442137  9.471854
## [30,]  9.972344 16.14122 11.486550 10.923891 10.912497
## [31,] 11.076405 14.76777 11.412524 11.319801 11.365783
## [32,]  9.548865 15.15986  9.548958  9.939631  9.941719
## [33,] 10.554616 18.17405 11.068023 10.128497 10.206939
## [34,]  9.644095 16.37363 12.425353 11.486175 11.436919
## [35,]  8.911594 12.85324 10.077476  9.826222  9.807718
## [36,] 10.274688 16.53689  9.919621 10.450361 10.420028
## [37,] 10.321125 22.15183 11.302090 12.144318 12.093851
## [38,] 10.474988 12.79161 11.376314 10.995788 10.949953
## [39,] 11.196118 14.50464 11.612124 10.761697 10.735767
## [40,]  9.377959 13.19416 10.755188 10.221776 10.173732
## [41,]  9.863707 12.49787  9.609476  9.167464  9.167795
## [42,]  9.342324 16.04264  9.907436  9.807772  9.753451
## [43,] 10.766856 14.17832  9.738652  9.565872  9.571354
## [44,] 10.317183 21.15646  9.247791  9.241420  9.256170
## [45,] 11.346331 17.17574 12.098131 12.024099 12.008611
## [46,] 12.125870 15.17440 12.314121 11.846624 11.779994
## [47,] 11.705125 25.37842 14.886763 13.768619 13.876279
## [48,]  9.608598 11.41428 10.180209  9.893366  9.876041
## [49,]  9.575885 12.25762 10.164216  9.660480  9.612535
## [50,] 10.672752 16.64013 11.901728 11.805444 11.702304
```

```
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
## 30.28367 29.50343 29.41676 29.63116 29.68231 29.68189
```

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

```
## [1] -0.7802395 -0.8669094 -0.6525141 -0.6013655 -0.6017858
```

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

```
## [1] 2.198867 2.904227 2.434833 2.265251 2.267683
```

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

```
## [1] 0.96 0.98 0.94 0.94 0.94
```

```r
Avg_ci_len_k0 = colMeans(result_k0$confidence_interval_length_matrix)
Avg_ci_len_k0
```

```
## [1] 10.17238 15.17260 10.76370 10.59966 10.58213
```

```r
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, Avg_ci_len = Avg_ci_len_k0)
```

Then we try to run under different k.

```r
#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 <- confidence_interval_length_matrix <- matrix(NA, nrow = M, ncol = num_

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)
  # Get Confidence Interval Length
  confidence_interval_lengths <- confidence_intervals[2, ] - confidence_intervals[1, ]
```

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

result_k0.05 <- list(true_tates, tate_matrix, coverage_matrix, confidence_interval_length_matrix)
names(result_k0.05) <- c("true_tates", "tate_matrix", "coverage_matrix", "confidence_interval_length_ma
#toc() #193.06 sec elapsed
```

Display result at k=0.05

```
result_k0.05
```

```
## $true_tates
##  [1] 28.45105 24.87642 28.26447 28.36134 32.22011 31.50874 29.16673 31.32175
##  [9] 25.54649 29.85141 27.71186 32.30022 30.48105 26.52799 32.32812 26.64736
## [17] 27.80385 30.24336 28.83708 30.24297 27.32281 28.95551 31.14651 29.72892
## [25] 28.22041 30.13611 29.39582 32.28305 29.90157 31.06297 24.05425 27.49465
## [33] 30.65595 27.69503 29.75060 29.80316 31.11132 33.62069 33.18279 30.58718
## [41] 31.27822 27.61269 27.10214 29.06886 30.05420 27.67567 29.45863 29.89659
## [49] 31.52874 29.11092
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 30.48421 25.04110 28.12836 28.73461 28.73706
##  [2,] 30.27574 30.21362 29.58980 29.64679 29.64638
##  [3,] 31.49823 31.85087 31.64901 31.56851 31.56807
##  [4,] 31.91862 28.29131 32.69937 32.54592 32.53671
##  [5,] 30.41871 31.94440 30.92795 30.74932 30.75475
##  [6,] 34.71904 28.27531 34.01698 34.31754 34.27503
##  [7,] 33.53647 32.75851 34.72867 34.73776 34.73095
##  [8,] 31.22222 33.86401 31.89603 31.86403 31.87586
##  [9,] 24.96260 28.35353 26.06264 25.79613 25.79399
## [10,] 31.44731 24.47384 36.16653 35.41883 35.31618
## [11,] 30.65936 27.85464 30.22380 30.45257 30.45320
## [12,] 27.21331 29.20917 28.28579 28.35830 28.34088
## [13,] 33.41288 28.99781 32.45988 31.88729 31.88387
## [14,] 30.11058 33.91731 30.72407 30.41252 30.41595
## [15,] 28.39523 25.86691 24.30436 24.62189 24.73349
## [16,] 28.45381 23.13215 26.53754 26.36918 26.36682
## [17,] 27.59555 24.82542 28.00968 27.53922 27.54139
## [18,] 28.43005 27.37796 27.96759 28.02714 28.02439
## [19,] 31.28389 37.20254 32.32646 32.15150 32.14923
## [20,] 32.08961 32.86736 32.66509 32.64522 32.64290
## [21,] 32.47389 32.90372 33.14547 33.02166 33.01033
## [22,] 21.95095 23.07871 21.45950 21.88828 21.88675
## [23,] 26.05677 23.68569 25.37953 25.74823 25.74612
## [24,] 32.06890 33.81382 31.47708 31.69645 31.69621
## [25,] 31.78626 32.59870 31.43383 31.55301 31.55692
## [26,] 32.96792 32.72150 31.79736 32.71774 32.71922
## [27,] 30.07451 26.37828 29.27825 29.22195 29.23301
```

```
## [28,] 29.31964 29.45984 29.67162 29.90892 29.90864
## [29,] 31.93044 28.44965 30.00607 30.02704 30.00619
## [30,] 27.71141 25.20756 27.36984 26.75012 26.73991
## [31,] 29.12070 31.50922 31.32688 31.43390 31.38303
## [32,] 31.97765 34.68200 31.11518 31.31623 31.33448
## [33,] 26.94716 28.80737 27.22890 27.40154 27.39051
## [34,] 28.29374 25.01927 29.53701 29.32480 29.32585
## [35,] 28.60644 30.57079 28.79247 29.66648 29.66355
## [36,] 29.84891 27.55407 30.30272 29.70034 29.71341
## [37,] 29.74123 28.50686 28.41947 28.66895 28.67436
## [38,] 30.30396 29.41496 30.05571 29.89199 29.90944
## [39,] 32.53683 33.46556 34.14160 33.75382 33.73474
## [40,] 29.01226 28.20161 29.28234 29.08342 29.07722
## [41,] 38.73782 37.89764 40.78622 40.65956 40.66810
## [42,] 25.94525 28.73343 26.63151 26.39766 26.40457
## [43,] 28.58767 27.93296 28.05616 27.96397 27.96997
## [44,] 32.38999 34.29524 33.66579 33.46474 33.46369
## [45,] 28.20203 30.00219 29.49599 29.21686 29.22433
## [46,] 30.89099 23.51866 29.60560 29.53244 29.54791
## [47,] 30.43936 30.93350 30.03481 30.19272 30.19303
## [48,] 32.44599 28.82728 31.43757 31.21598 31.27532
## [49,] 30.67328 35.31362 28.87850 28.58434 28.55620
## [50,] 26.90173 28.13892 25.99566 26.13085 26.13217
##
## $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    0    0
##  [8,]    1    1    1    1    1
##  [9,]    1    1    1    1    1
## [10,]    1    1    0    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    0    0
## [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    1    0    0    0
## [22,]    0    0    0    0    0
## [23,]    0    0    0    0    0
## [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,]    1    1    1    1    1
```

```
## [29,]    1    1    1    1    1
## [30,]    1    1    1    1    1
## [31,]    1    1    0    0    0
## [32,]    0    0    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,]    0    1    0    0    0
## [42,]    1    1    1    1    1
## [43,]    1    1    1    1    1
## [44,]    1    0    1    1    1
## [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
##
## $confidence_interval_length_matrix
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,]  9.906277 10.874600  9.053676  9.406751  9.412631
##  [2,] 11.196442 18.696870 11.723829 12.024765 12.011390
##  [3,] 10.221164 12.583237 10.245630  9.932418  9.926983
##  [4,] 10.194936 16.913288  9.385151  9.735278  9.766591
##  [5,] 10.359240 13.416563 10.574489 11.535915 11.540206
##  [6,] 10.410246 15.515911 13.501308 14.400679 14.506481
##  [7,]  9.246052 12.919321 11.216857 10.264227 10.267363
##  [8,] 10.942824 16.935765 10.435935 10.703411 10.701622
##  [9,]  9.779165 16.927582 12.687278 12.061844 12.023775
## [10,]  9.777845 19.186117 14.378253 13.402486 13.026896
## [11,]  8.427718  9.495773  9.111033  8.945165  8.934268
## [12,] 11.269133 18.609562 12.730392 12.451378 12.407325
## [13,] 10.044446 10.970997 10.757044 10.076426 10.076420
## [14,] 10.121809 21.880455 11.201415 10.781803 10.785162
## [15,] 11.757474 21.897109 15.589693 14.129346 13.803773
## [16,] 12.138065 16.951729 14.426591 13.985905 13.916520
## [17,] 10.484891 11.734744 11.077440 10.799257 10.776492
## [18,] 10.784995 11.418405 11.740131 11.770817 11.775385
## [19,]  8.471075 15.379537  9.922494  9.446129  9.403766
## [20,]  9.405841 10.783456  9.885069  9.644919  9.643119
## [21,]  8.829028 20.212264 11.092513 10.474719 10.347593
## [22,]  8.807879  9.876226  8.713352  8.776146  8.775291
## [23,]  8.731669 14.693172  9.386722  8.540860  8.529567
## [24,] 12.027098 22.048071 13.374816 12.513447 12.489904
## [25,]  9.323483 11.341166 11.968953 10.780642 10.762688
## [26,]  9.707836 21.779865 11.358590  9.467817  9.448655
## [27,]  9.326612 13.772054  9.428936  9.699145  9.670618
## [28,]  9.419627 14.400085 11.788193 12.016219 12.029304
## [29,]  9.635852 16.164301 10.838103 10.800486 10.807780
```

```
## [30,]  9.573844 17.958243 10.862693 11.028999 11.057558
## [31,] 11.774755 21.019868 12.809870 12.610492 12.627950
## [32,]  9.860231 10.097133  9.936860  9.630585  9.620584
## [33,]  8.638760 15.821453 11.891166 11.741276 11.700042
## [34,] 10.982122 19.502110 12.651276 12.381021 12.362772
## [35,] 10.842452 16.106719 10.802109 11.715919 11.732344
## [36,]  9.577154 12.637496  9.644395  9.525096  9.552764
## [37,]  8.710316 12.339413 10.110631  9.392960  9.377148
## [38,] 10.589815 21.942934 11.967388 11.601260 11.492213
## [39,]  9.936467 18.684435 13.676840 13.201635 13.112094
## [40,]  8.966568 12.451736 10.105780 10.305196 10.272416
## [41,] 11.669053 19.650308 13.696360 12.400464 12.434726
## [42,]  9.222697 16.398353  9.584373  9.608772  9.664947
## [43,] 10.785986 11.194498 11.526624 11.240511 11.231470
## [44,]  8.968966  9.509980  9.801119  9.006886  8.998122
## [45,]  9.559957 12.751847 10.295609 10.169979 10.188536
## [46,]  8.906970 18.508751  9.832519 10.033359 10.142012
## [47,]  8.769428 10.845907  8.596958  8.544386  8.545956
## [48,] 11.002409 22.697716 16.118863 16.392774 16.156970
## [49,] 12.807228 19.305457 13.196652 13.269940 13.261908
## [50,] 10.717334 16.006770 12.266077 11.333685 11.319688
```

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
## 29.43177 30.12142 29.55881 30.10357 30.07957 30.07865
```

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

```
## [1] 0.6896554 0.1270415 0.6717988 0.6477986 0.6468794
```

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

```
## [1] 2.700872 3.385699 2.870818 2.819693 2.814896
```

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

```
## [1] 0.90 0.92 0.88 0.86 0.86
```

```
Avg_ci_len_k0.05 = colMeans(result_k0.05$confidence_interval_length_matrix)
Avg_ci_len_k0.05
```

```
## [1] 10.05222 15.65619 11.33936 11.07407 11.04840
```

```
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, Avg_ci_len = Avg_ci_len_k0.05)
```

I also tried a different k

```
#tic()
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 <- confidence_interval_length_matrix <- matrix(NA, nrow = M, ncol = num_

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)
  # Get Confidence Interval Length
  confidence_interval_lengths <- confidence_intervals[2, ] - confidence_intervals[1, ]

  # Store outputs
  true_tates[m] <- tate_true_m
  tate_matrix[m, ] <- tates_m
  coverage_matrix[m, ] <- coverage_indicators
  confidence_interval_length_matrix[m, ] <- confidence_interval_lengths
}
#toc() #206.86 sec elapsed
result_k0.2 <- list(true_tates, tate_matrix, coverage_matrix, confidence_interval_length_matrix)
names(result_k0.2) <- c("true_tates", "tate_matrix", "coverage_matrix", "confidence_interval_length_mat
```

Display result at k=0.2

```
result_k0.2
```

```
## $true_tates
```

```
##  [1] 32.67235 31.71693 33.26653 26.81768 30.77060 28.59148 29.66329 33.10225
##  [9] 29.47096 28.24409 28.87826 33.19723 31.44773 30.80044 30.49412 33.69762
## [17] 28.80561 30.98326 30.36366 26.59354 29.87017 29.74180 32.95768 28.37237
## [25] 26.77255 30.65276 27.89112 29.64254 28.13926 29.35322 30.05738 29.44314
## [33] 32.00448 29.95206 29.55995 29.46022 27.19813 29.67804 31.98291 31.07076
## [41] 32.93417 29.47567 32.31344 29.70186 30.95416 30.22185 30.25995 30.40469
## [49] 31.55360 30.21787
##
## $tate_matrix
##           [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 29.93654 29.52854 26.82415 27.82961 27.83978
##  [2,] 31.99985 32.67473 34.57102 34.48219 34.50637
##  [3,] 29.10051 28.68169 28.41899 28.40633 28.41237
##  [4,] 26.70353 36.82412 28.65221 28.25287 28.24520
##  [5,] 25.91776 28.82029 22.40796 22.24003 22.32361
##  [6,] 30.09496 25.94352 30.13296 30.12495 30.11227
##  [7,] 30.72181 32.20696 31.10922 30.72143 30.72588
##  [8,] 30.61098 30.63722 30.46825 30.26614 30.26858
##  [9,] 31.02315 28.85600 31.13847 31.80869 31.81415
## [10,] 26.29098 29.23168 25.38788 26.58808 26.60104
## [11,] 32.25504 36.93775 32.16038 32.14652 32.14674
## [12,] 30.93734 39.17221 33.87011 33.15201 33.23586
## [13,] 31.98507 31.43841 29.85661 30.38742 30.38352
## [14,] 33.01356 36.23115 33.47742 33.86801 33.86708
## [15,] 30.66655 25.84905 27.26649 28.50422 28.49001
## [16,] 31.98740 28.70917 31.29740 31.26772 31.28912
## [17,] 29.77989 31.36448 30.58534 30.65361 30.64434
## [18,] 29.49357 25.19086 27.74725 27.63437 27.62590
## [19,] 29.48222 33.96264 28.70982 29.36252 29.37878
## [20,] 20.89143 32.79308 23.35520 23.29087 23.31584
## [21,] 29.81987 33.51244 29.91727 30.56636 30.56913
## [22,] 27.92888 31.47004 28.73710 28.65132 28.64723
## [23,] 24.82034 22.90257 25.96010 25.57321 25.54230
## [24,] 30.46157 34.25117 29.44850 29.05111 29.13100
## [25,] 31.42174 28.84502 30.70062 30.32340 30.33673
## [26,] 30.45935 35.01729 28.94340 29.79922 29.79782
## [27,] 27.87181 27.97628 27.50808 27.70853 27.70981
## [28,] 23.66615 25.43201 22.39886 22.72762 22.74697
## [29,] 29.28064 24.03898 31.05202 30.52321 30.47786
## [30,] 31.49940 32.66172 28.88600 29.07006 29.08230
## [31,] 27.07716 22.98401 24.27338 24.61201 24.61786
## [32,] 31.59681 40.02051 34.40714 34.79933 34.69273
## [33,] 27.32348 26.49435 28.29219 28.66355 28.64463
## [34,] 25.62504 28.15253 27.32746 27.66571 27.64862
## [35,] 30.44005 28.15217 29.95530 29.80351 29.80048
## [36,] 30.28732 28.37552 29.76522 29.54158 29.55230
## [37,] 29.00660 33.81938 31.14058 30.99592 30.94812
## [38,] 29.80520 34.30937 30.22495 30.13090 30.14526
## [39,] 34.53779 32.29952 34.19611 34.38314 34.38627
## [40,] 26.65453 26.50543 27.50112 27.38439 27.37828
## [41,] 37.24675 35.54448 37.37180 37.68462 37.66524
## [42,] 25.71443 26.83055 26.82572 26.80777 26.80675
## [43,] 34.07945 35.34056 36.34760 36.43182 36.47968
## [44,] 27.17227 27.20881 27.86122 27.69660 27.69019
```

```
## [45,] 29.20073 32.53360 31.15967 31.33742 31.31302
## [46,] 31.88006 32.73227 31.81120 32.11854 32.11712
## [47,] 26.24375 26.65473 27.53863 27.18146 27.17944
## [48,] 29.24453 31.45173 28.58253 28.86867 28.87033
## [49,] 31.50099 30.88869 31.90033 30.92864 30.93373
## [50,] 23.83605 24.69516 24.09301 24.05533 24.05078
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1    1    0    0    0
##  [2,]    1    1    1    1    1
##  [3,]    1    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    0    0    0
##  [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    0    1    1    1
## [12,]    1    1    1    1    1
## [13,]    1    1    1    1    1
## [14,]    1    0    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,]    1    1    1    1    1
## [22,]    1    1    1    1    1
## [23,]    0    0    0    0    0
## [24,]    1    1    1    1    1
## [25,]    0    1    1    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,]    1    1    1    1    1
## [31,]    1    0    0    1    1
## [32,]    1    1    1    1    1
## [33,]    0    0    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    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    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,]    0    1    0    0    0
##
## $confidence_interval_length_matrix
##            [,1]      [,2]      [,3]      [,4]      [,5]
##  [1,]  7.932764 10.427710 10.298740  8.933167  8.901515
##  [2,] 11.925516 23.136456 14.067226 14.053150 14.020462
##  [3,] 11.469319 16.105428 12.419024 11.797415 11.818218
##  [4,] 12.243937 29.705632 18.633212 19.042754 18.795657
##  [5,] 12.525121 22.734412 18.280200 17.366220 17.231891
##  [6,]  9.155770 14.329781 11.176971 10.389715 10.341485
##  [7,]  8.764620 11.760308  9.144024  9.015100  8.995683
##  [8,]  9.784347 12.385733 10.110025  9.886054  9.918449
##  [9,]  9.391187  9.576752  9.339533  9.560371  9.583147
## [10,] 13.194571 14.194910 11.762163 11.798518 11.816619
## [11,] 10.874421 14.294912 12.082000 12.297829 12.271106
## [12,] 14.064358 24.409617 13.377921 15.769078 15.510189
## [13,] 12.542604 24.388455 13.245428 13.570477 13.614314
## [14,]  9.667591 10.256369  9.250505  9.156866  9.151331
## [15,] 10.742265 20.777812 14.399869 14.255715 14.303303
## [16,] 11.550429 27.160764 14.840929 15.029104 14.741795
## [17,] 10.791815 12.759642 10.261054 10.551873 10.547310
## [18,] 14.242175 28.417315 15.885608 15.666538 15.702546
## [19,] 10.983881 15.238635 11.060024 10.926671 10.922650
## [20,] 14.741721 23.416877 15.022106 15.665066 15.579698
## [21,]  9.797944 15.363046 10.294217 10.682509 10.699873
## [22,]  9.536024 14.659084 10.827773  9.828816  9.786199
## [23,] 10.017182 15.918649 11.000036 10.288363 10.253009
## [24,] 12.700916 25.041205 13.342079 12.709172 12.610914
## [25,]  9.136660 12.468439  8.839727  9.076767  9.054953
## [26,] 10.746538 15.311110 10.718488 10.983519 10.982515
## [27,]  8.747345 12.037928  9.865929 10.123003 10.112538
## [28,] 11.838309 32.131160 16.226526 14.527463 14.302790
## [29,] 13.125872 28.384850 13.275986 14.098774 13.921468
## [30,]  9.411231 14.622778 11.110457 10.552575 10.457659
## [31,] 10.381678 14.314449 11.387948 11.960008 11.980354
## [32,] 14.651059 44.005493 18.071024 17.609026 17.258664
## [33,]  9.514470 11.909811 10.210273  9.980539  9.957893
## [34,] 11.746183 17.670914 13.125282 13.022220 12.971429
## [35,]  9.445507 10.748377 10.837193 10.246770 10.237280
## [36,]  9.428928 16.447920 11.822159 10.932103 10.985069
## [37,] 11.384091 23.083265 12.103746 11.742124 11.683396
## [38,] 13.969250 26.729086 17.188746 17.221807 17.205203
## [39,] 12.831447 15.906813 13.391224 13.236384 13.218326
## [40,]  9.014461 10.036068  9.197383  9.385416  9.388735
## [41,] 12.112349 25.851239 12.863241 12.986881 12.850313
## [42,]  9.151312 11.941382  9.343227  9.045581  9.061552
## [43,] 11.608727 23.194147 12.502944 13.653887 13.590741
## [44,]  8.944213 12.578520 10.519324 10.345614 10.309348
## [45,] 10.975907 21.527781 13.824174 14.501012 14.387822
## [46,]  9.442310 12.771925  9.781521 10.484545 10.470980
```

```
## [47,] 10.276687 13.553661 11.241274 10.943637 10.918950
## [48,]  8.264410 11.946409  8.495313  8.311462  8.289851
## [49,]  9.952723 13.414629 10.727851  9.902888  9.859829
## [50,]  9.010825 13.914407  9.022713  9.273782  9.239010
```

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
## 30.22831 29.37190 30.52309 29.43128 29.52145 29.52377
```

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

```
## [1] -0.8564108  0.2947802 -0.7970244 -0.7068579 -0.7045395
```

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

```
## [1] 2.367818 3.717977 2.745008 2.631023 2.627364
```

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

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

```
Avg_ci_len_k0.2 = colMeans(result_k0.2$confidence_interval_length_matrix)
Avg_ci_len_k0.2
```

```
## [1] 10.87506 18.05924 12.11629 12.04777 11.99628
```

```
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, Avg_ci_len = Avg_ci_len_k0.2)
```

```
#tic()
k <- 0.5

# Matrix for coverage indicators
M <- 50

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

set.seed(seed)
```

```r
true_tates <- rep(NA, M)
tate_matrix <- coverage_matrix <- confidence_interval_length_matrix <- matrix(NA, nrow = M, ncol = num_

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)
  # Get Confidence Interval Length
  confidence_interval_lengths <- confidence_intervals[2, ] - confidence_intervals[1, ]

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

result_k0.5 <- list(true_tates, tate_matrix, coverage_matrix, confidence_interval_length_matrix)
names(result_k0.5) <- c("true_tates", "tate_matrix", "coverage_matrix", "confidence_interval_length_mat
#toc() #215.7 sec elapsed
```

Display result at k=0.5

```r
result_k0.5
```

```
## $true_tates
##  [1] 28.36157 27.50219 28.39830 27.44625 27.82908 33.06531 30.67735 31.69054
##  [9] 30.07537 27.14704 28.78037 29.33137 27.93617 32.06727 29.24920 31.31908
## [17] 29.68258 30.69466 32.66857 29.16996 30.82056 28.83853 27.70075 28.77104
## [25] 29.77264 27.61514 32.99485 28.33534 34.28900 30.50399 29.58656 29.46058
## [33] 29.13964 29.68072 26.11483 30.27073 30.56883 31.70495 31.49292 31.59065
## [41] 27.62617 30.04509 28.84669 30.30085 27.18015 31.82906 29.56550 29.88816
## [49] 29.29287 29.37241
##
## $tate_matrix
##           [,1]      [,2]     [,3]     [,4]     [,5]
##  [1,] 37.87576 42.882640 37.09317 36.89570 36.94737
##  [2,] 21.66481 12.668148 21.17577 22.32184 22.18035
##  [3,] 39.23642 35.052382 39.53489 39.10356 39.12510
##  [4,] 27.30128 32.712104 22.92784 24.14528 23.88312
##  [5,] 32.97191 36.058937 31.20718 30.97948 30.98447
```

```
##  [6,] 33.76480 35.297854 32.83086 32.43600 32.48100
##  [7,] 22.19420 27.843618 26.33087 25.44673 25.52599
##  [8,] 23.31443 24.838999 22.80556 23.10625 23.11017
##  [9,] 26.37462 35.096086 26.01095 26.99983 27.08577
## [10,] 20.15701 28.347195 22.34439 22.34761 22.38524
## [11,] 25.55994 27.932909 26.81481 26.32656 26.30897
## [12,] 39.06511 34.282673 39.21026 39.62701 39.62820
## [13,] 37.64678 36.508525 37.98861 37.81608 37.82271
## [14,] 32.41285 33.220225 36.79527 37.15333 36.92819
## [15,] 34.23139 34.560720 34.49237 34.36944 34.37276
## [16,] 34.74270 41.244932 35.05135 34.94206 34.91444
## [17,] 30.01019 33.518035 28.08336 28.15551 28.14829
## [18,] 33.18686 38.218667 36.24120 36.79267 36.71402
## [19,] 29.09256 31.375024 23.50191 23.42085 23.71225
## [20,] 23.41727 22.807018 23.55116 23.17974 23.17727
## [21,] 35.60603 36.677034 34.68834 34.70065 34.70662
## [22,] 38.18708 46.681240 46.95562 44.95819 44.58459
## [23,] 20.85116 18.452591 20.38239 21.54219 21.56157
## [24,] 38.67914 38.749678 39.90962 40.26384 40.24340
## [25,] 23.20526 20.441078 23.25342 23.27938 23.24559
## [26,] 23.83850 20.765010 25.25457 24.86527 24.81645
## [27,] 28.29595 30.659249 26.54431 26.74917 26.75154
## [28,] 29.30650 30.561571 30.19307 30.00562 30.00008
## [29,] 25.17791 23.251989 26.89818 26.37496 26.36386
## [30,] 24.95924 26.348789 24.88017 24.51346 24.51142
## [31,] 29.55149 31.932977 31.21826 31.12235 31.09164
## [32,] 30.83265 38.808545 31.35276 31.86304 31.91046
## [33,] 19.54254  2.114486 14.83329 14.30111 14.64709
## [34,] 25.61475 22.771872 17.21518 18.36654 18.53700
## [35,] 28.60772 17.247780 17.04562 18.06349 18.20010
## [36,] 22.64176 16.920589 19.86179 19.27480 19.33239
## [37,] 40.37807 32.703920 39.21168 40.04326 40.08277
## [38,] 32.02605 39.298256 33.51609 35.23209 35.12064
## [39,] 31.35536 40.972753 32.83864 32.36297 32.38239
## [40,] 24.36711 17.231903 24.71393 24.78573 24.80007
## [41,] 38.46039 36.860958 36.78905 36.30443 36.33357
## [42,] 35.67357 33.903744 35.26155 35.32237 35.31964
## [43,] 34.74906 29.827811 33.83922 33.50298 33.51912
## [44,] 38.84530 38.979659 41.01284 40.97103 40.95482
## [45,] 35.54102 40.002662 35.98894 35.88827 35.89810
## [46,] 38.08021 47.778977 37.60815 37.85286 37.64664
## [47,] 25.47420 26.362814 26.65863 27.34610 27.20387
## [48,] 32.65745 44.154707 35.40238 35.01465 35.05207
## [49,] 25.44923 21.723461 26.06810 25.34481 25.34739
## [50,] 29.25761 27.622595 27.04776 26.99182 27.03110
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    0    0    0    0    0
##  [2,]    1    1    1    1    1
##  [3,]    0    1    0    0    0
##  [4,]    1    1    1    1    1
##  [5,]    1    0    1    1    1
##  [6,]    1    1    1    1    1
```

```
##  [7,]    0    1    1    1    1
##  [8,]    0    0    0    0    0
##  [9,]    1    1    1    1    1
## [10,]    0    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    0    1    1    0    0
## [13,]    0    0    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,]    0    1    1    0    0
## [21,]    1    1    1    1    1
## [22,]    0    0    0    0    0
## [23,]    1    1    1    1    1
## [24,]    0    1    0    0    0
## [25,]    0    1    1    1    1
## [26,]    1    1    1    1    1
## [27,]    0    1    0    0    0
## [28,]    1    1    1    1    1
## [29,]    0    0    0    0    0
## [30,]    1    1    1    1    1
## [31,]    1    1    1    1    1
## [32,]    1    1    1    1    1
## [33,]    0    0    0    0    0
## [34,]    1    1    0    1    1
## [35,]    1    1    1    1    1
## [36,]    0    0    0    0    0
## [37,]    0    1    1    1    1
## [38,]    1    1    1    1    1
## [39,]    1    0    1    1    1
## [40,]    0    0    1    1    1
## [41,]    0    0    0    0    0
## [42,]    0    1    0    0    0
## [43,]    0    1    1    1    1
## [44,]    0    1    0    0    0
## [45,]    0    0    0    0    0
## [46,]    1    1    1    1    1
## [47,]    1    1    1    1    1
## [48,]    1    0    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    1    1    1    1
##
## $confidence_interval_length_matrix
##            [,1]     [,2]      [,3]      [,4]      [,5]
##  [1,] 10.196134 20.11457 12.571721 13.483035 13.261826
##  [2,] 15.673016 48.66601 21.359077 24.140428 23.706144
##  [3,] 11.193201 17.74668 12.664918 12.098683 12.117980
##  [4,] 21.213108 39.73008 22.424185 21.251041 21.271609
##  [5,] 10.335921 15.54771 10.692568 10.670523 10.598816
##  [6,] 10.288783 15.09505 10.378015  9.592030  9.541062
##  [7,] 12.712099 22.44545 14.717041 15.871513 15.965759
```

```
##  [8,] 10.169344 12.54682 11.142337 10.024066  9.987554
##  [9,] 14.223367 22.21582 15.743596 15.351596 15.315046
## [10,] 12.856868 25.98423 18.139946 17.367500 17.336490
## [11,] 18.457021 41.78358 25.867745 24.635024 24.249811
## [12,] 14.372266 31.99546 18.024464 16.890463 16.948263
## [13,] 10.967909 18.49350 10.799004 10.404588 10.439190
## [14,] 16.825449 44.39594 20.687239 19.978253 19.888053
## [15,] 13.559116 22.53121 14.390244 14.686901 14.743350
## [16,] 13.071409 27.20641 14.154769 14.289047 14.480235
## [17,] 13.094207 27.39506 15.603645 14.869191 14.778772
## [18,] 14.925976 37.67198 17.985234 18.696208 18.218108
## [19,] 20.047268 58.41211 27.261889 27.382858 25.234361
## [20,]  9.666893 15.89960 11.067591 10.816219 10.770745
## [21,] 11.050540 23.67893 11.524147 12.224489 12.267435
## [22,] 12.747396 35.14365 17.014650 16.513937 15.783316
## [23,] 19.641095 30.56203 22.522811 22.115327 22.092901
## [24,] 11.692544 27.73318 15.122372 14.866566 14.598877
## [25,] 13.598753 42.90882 22.231344 23.381931 22.133481
## [26,] 16.165009 36.60049 19.611116 18.739688 18.811838
## [27,]  8.455078 11.07828  8.837976  8.542940  8.499830
## [28,] 14.811068 28.86811 14.422670 15.900828 16.105519
## [29,]  9.945726 12.75342 10.543192 10.673170 10.683773
## [30,] 12.791942 19.78648 12.188869 12.060900 12.080018
## [31,] 13.442787 37.67659 20.047942 19.996463 19.420516
## [32,] 14.940750 25.48872 14.198826 15.232382 15.228173
## [33,] 13.982849 48.73769 19.133848 20.632202 19.868988
## [34,] 16.941582 44.72110 25.720512 27.992568 27.371853
## [35,] 19.532449 54.08518 25.529740 25.522153 24.477407
## [36,]  9.850956 20.89664 12.774493 12.312792 12.385994
## [37,] 18.771746 44.03914 21.473204 21.112926 21.008507
## [38,] 15.512652 41.27762 16.331149 15.823504 15.909196
## [39,] 14.350195 19.86697 16.682361 16.462590 16.452478
## [40,] 13.726234 20.02623 16.929484 15.507045 15.376008
## [41,] 11.395795 16.93521 13.375306 13.009615 12.918838
## [42,]  9.915439 10.91570  9.091364  9.305828  9.291847
## [43,] 11.255432 23.74964 14.273120 13.687979 13.482530
## [44,] 12.132403 22.45916 11.429336 12.145607 12.035918
## [45,]  8.927291 14.14566  8.752733  9.122677  9.107375
## [46,] 20.139621 44.87933 21.072268 22.091280 21.928945
## [47,] 16.079557 48.14563 18.267058 19.188611 18.373873
## [48,] 15.155970 24.64763 18.212998 16.050362 15.863659
## [49,] 13.597503 20.53883 14.975706 15.499975 15.570332
## [50,] 20.251259 36.40635 21.437606 21.779305 21.494404
```

```
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
## 29.72583 30.22866 30.88551 30.00871 30.05546 30.05263
```

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

```
## [1] 0.5028351 1.1596790 0.2828783 0.3296301 0.3268056
```

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

```
## [1] 5.277548 7.520190 6.233855 6.174599 6.145260
```

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

```
## [1] 0.56 0.74 0.70 0.68 0.68
```

```
Avg_ci_len_k0.5 = colMeans(result_k0.5$confidence_interval_length_matrix)
Avg_ci_len_k0.5
```

```
## [1] 13.89302 29.09259 16.38807 16.39990 16.18954
```

```
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, Avg_ci_len = Avg_ci_len_k0.5)
```

```
#tic()
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 <- confidence_interval_length_matrix <- matrix(NA, nrow = M, ncol = num_

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)
  # Get Confidence Interval Length
  confidence_interval_lengths <- confidence_intervals[2, ] - confidence_intervals[1, ]

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

result_k0.8 <- list(true_tates, tate_matrix, coverage_matrix, confidence_interval_length_matrix)
names(result_k0.8) <- c("true_tates", "tate_matrix", "coverage_matrix", "confidence_interval_length_mat
#toc() #223.56 sec elapsed
```

```
result_k0.8
```

```
## $true_tates
##  [1] 29.66270 33.50603 29.55749 30.21253 31.82892 28.40739 30.89844 32.64921
##  [9] 26.53674 29.92930 28.46513 27.34995 31.58493 28.95969 28.02848 28.21879
## [17] 29.95082 30.52091 28.82475 32.28849 28.88109 28.73448 30.46662 31.68017
## [25] 30.32322 30.80871 32.58305 27.19843 31.31085 30.16944 31.90807 33.34541
## [33] 26.94607 29.88520 30.40959 28.21462 28.82566 30.28378 29.23919 29.18354
## [41] 28.88032 31.61830 28.10520 30.20252 26.29674 29.35748 27.22940 30.14588
## [49] 30.79891 29.71784
##
## $tate_matrix
##           [,1]      [,2]      [,3]      [,4]     [,5]
##  [1,] 16.15394 30.207266 26.066140 25.786098 25.12998
##  [2,] 20.11050 17.342972 29.402539 28.527309 28.46652
##  [3,] 19.67271  5.621414 18.804771 18.132933 17.64161
##  [4,] 24.37203 33.560064 20.714831 21.783901 21.85720
##  [5,] 27.29406 24.642784 26.623671 26.386623 26.48270
##  [6,] 35.35211 70.835222 31.695133 36.467213 34.16326
##  [7,] 33.03324 -6.260083 36.140002 33.423164 33.47179
##  [8,] 28.28977 20.948477 16.558643 19.570880 18.97053
##  [9,] 43.79941 35.319226 39.848518 41.363164 41.26462
## [10,] 13.93058 17.593732 16.534222 15.217961 15.44543
## [11,] 32.63209 23.632266 28.310932 28.699474 28.71748
## [12,] 18.99826 36.032380 22.291224 18.341207 18.76388
## [13,] 28.69884 18.273033 30.316175 29.478603 29.35984
## [14,] 46.02851 58.300224 48.313368 50.463878 50.46629
## [15,] 45.43451 45.385609 47.987889 47.048148 47.06904
## [16,] 17.98844  7.349228 10.669419 11.676084 11.33053
## [17,] 28.58466 13.687671 34.265519 32.038791 32.14815
## [18,] 19.57406 27.978793 22.148391 21.056927 21.19110
## [19,] 12.86840 18.051514  7.354574  7.864736  7.74512
## [20,] 26.00613 -2.169100 22.650862 23.692676 22.93166
## [21,] 40.83587 41.821874 39.204802 36.146066 36.40516
```

```
## [22,] 27.15923 26.248907 15.929192 15.566505 15.65489
## [23,] 55.40350 69.353819 69.273230 69.078413 69.83283
## [24,] 37.53180 57.729454 41.732477 40.482568 40.63947
## [25,] 17.78147 27.497834 15.521991 15.755717 15.81758
## [26,] 42.14650 55.874350 51.481512 51.113276 50.78972
## [27,] 20.31688 26.589154 23.624515 23.996507 23.96289
## [28,] 36.64426 45.643860 37.857145 38.593432 38.54641
## [29,] 21.53582 10.162015 22.846880 21.255357 21.41453
## [30,] 21.18179 17.102150 18.936033 16.686831 16.64555
## [31,] 26.35740 45.030658 28.103339 28.124894 28.14765
## [32,] 16.54220 34.578735 26.702712 26.767195 26.84902
## [33,] 42.71432 35.806226 43.951126 43.018369 42.83458
## [34,] 20.86607 10.201499 17.222518 18.426162 18.48677
## [35,] 47.40613 62.220892 40.232382 45.486526 45.19285
## [36,] 41.18093 25.077756 35.921474 37.337623 36.88245
## [37,] 38.96615 34.694630 37.436574 38.258570 38.17558
## [38,] 11.29281 42.416315 11.493488 11.467369 11.77659
## [39,] 40.56622 34.930059 40.693968 41.891766 41.90542
## [40,] 16.78252  7.991457 21.645299 23.277202 22.61678
## [41,] 11.84576 13.675461 11.087647 11.194395 11.14388
## [42,] 27.95383 16.089347 28.532876 27.453423 27.38049
## [43,] 28.91637 11.869886 32.662691 33.114489 33.32599
## [44,] 28.24989 29.650541 27.140241 28.836004 28.79847
## [45,] 39.61817 23.195568 42.095346 43.800607 44.03498
## [46,] 36.58503 33.498878 32.594395 33.239498 33.24874
## [47,] 24.57557 54.726880 35.892927 34.659049 34.35414
## [48,] 17.24251 26.125968 21.109656 22.160917 22.28134
## [49,] 24.94506 24.267632 22.468265 22.876325 22.92287
## [50,] 23.24335 33.371462 18.707842 20.246244 20.11935
##
## $coverage_matrix
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    0    1    1    1    1
##  [2,]    0    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,]    0    1    0    0    0
## [10,]    0    1    1    1    1
## [11,]    1    1    1    1    1
## [12,]    1    1    1    1    1
## [13,]    1    1    1    1    1
## [14,]    0    0    0    0    0
## [15,]    0    1    0    0    0
## [16,]    1    1    0    0    0
## [17,]    1    1    1    1    1
## [18,]    0    1    1    1    1
## [19,]    0    1    0    0    0
## [20,]    1    0    1    1    1
## [21,]    0    1    1    1    1
## [22,]    1    1    1    1    1
```

```
## [23,]    0    0    0    0    0
## [24,]    1    1    1    1    1
## [25,]    0    1    0    0    0
## [26,]    0    1    0    0    0
## [27,]    0    1    1    1    1
## [28,]    0    0    0    0    0
## [29,]    0    0    1    0    0
## [30,]    0    1    0    0    0
## [31,]    1    1    1    1    1
## [32,]    0    1    1    1    1
## [33,]    0    1    0    0    0
## [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,]    0    1    0    0    0
## [39,]    0    1    1    1    1
## [40,]    0    1    1    1    1
## [41,]    0    1    0    0    0
## [42,]    1    1    1    1    1
## [43,]    1    1    1    1    1
## [44,]    1    1    1    1    1
## [45,]    0    1    0    0    0
## [46,]    1    1    1    1    1
## [47,]    1    1    1    1    1
## [48,]    0    1    1    1    1
## [49,]    1    1    1    1    1
## [50,]    1    1    0    0    0
##
## $confidence_interval_length_matrix
##           [,1]      [,2]     [,3]     [,4]     [,5]
##  [1,] 20.55920  49.11032 24.64489 27.10818 25.08847
##  [2,] 19.41800  53.22745 23.47698 19.23373 20.05148
##  [3,] 33.12930  79.78064 55.33890 58.43833 49.88925
##  [4,] 24.22229  81.81203 26.23692 27.35333 26.81065
##  [5,] 17.33086  54.61342 26.19994 21.40704 20.88593
##  [6,] 21.21495 117.05533 34.46473 37.99774 34.69103
##  [7,] 27.59187  98.78933 31.20023 34.09087 30.62111
##  [8,] 37.79367 121.14465 46.60413 42.88108 42.76076
##  [9,] 16.78762  32.49183 15.17072 15.92017 15.53766
## [10,] 22.10445  38.03009 28.63703 27.72460 28.82078
## [11,] 12.70623  31.63364 20.08123 19.45336 19.31642
## [12,] 28.09297  58.33105 35.03860 34.29532 34.18980
## [13,] 18.99957  37.51374 23.85558 24.46278 24.52236
## [14,] 17.24499  28.49618 17.24931 17.61828 17.55972
## [15,] 17.30239  69.89530 24.12045 26.72388 25.12177
## [16,] 23.08131  49.80274 25.35181 26.05194 25.93813
## [17,] 21.08655  57.23542 29.21630 28.62659 28.85565
## [18,] 16.02308  31.97393 22.59780 21.50256 21.54188
## [19,] 21.64375  46.70336 21.66230 19.61499 19.62797
## [20,] 25.10426  74.02286 25.34292 26.25399 24.75716
## [21,] 20.17638  58.68084 28.45936 27.10258 26.18375
## [22,] 41.26279  97.72902 51.07079 47.56620 45.27548
## [23,] 48.07912 107.54761 60.01205 62.32454 61.51769
```

```
## [24,] 29.48723  72.67587 35.63230 33.07646 32.81163
## [25,] 19.23250  38.71982 22.00166 22.40609 22.30763
## [26,] 16.85804  52.03347 22.80943 24.28846 23.61006
## [27,] 17.90138  37.13212 19.61197 19.14785 19.08079
## [28,] 16.02460  41.07523 19.14797 19.27657 19.18087
## [29,] 16.14536  35.56399 20.02704 20.83931 21.33032
## [30,] 16.05336  34.22691 20.69115 19.69115 19.69670
## [31,] 18.88334  70.21381 24.48311 23.57524 22.91525
## [32,] 18.20025  45.43826 28.98516 32.54440 31.32662
## [33,] 15.13899  39.87830 24.00747 22.66880 21.40469
## [34,] 24.27252  64.97600 33.13293 33.14209 31.68651
## [35,] 36.78650  82.92491 37.98731 39.85006 39.22788
## [36,] 40.28577  70.20427 47.61758 47.91167 48.37251
## [37,] 36.00207  65.99322 44.30018 41.65232 43.05806
## [38,] 22.23861  88.31774 28.90210 27.43889 26.70512
## [39,] 19.01214  64.09729 31.41161 29.89364 27.87034
## [40,] 18.96531  48.48059 28.76172 27.80219 26.93727
## [41,] 24.91668  44.10890 24.09698 22.79693 22.40733
## [42,] 19.40707  47.39753 23.28111 23.70370 23.56483
## [43,] 39.19640  73.80401 49.63834 44.88136 46.70188
## [44,] 21.75228  75.93951 25.38453 27.48027 26.02852
## [45,] 15.55050  43.14203 18.95075 19.05354 18.52383
## [46,] 15.75835  36.21281 15.09099 16.90266 16.76742
## [47,] 22.81605  85.20530 34.47912 38.97523 34.53485
## [48,] 20.69626  39.79662 28.06402 26.41416 26.62399
## [49,] 21.42411  54.40811 27.27274 28.17815 28.11169
## [50,] 17.62435  41.02743 19.26427 18.24387 18.36303
```

```
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
## 29.80261 28.50419 29.47552 28.97599 29.14662 29.05607
```

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

```
## [1] -1.2984163 -0.3270902 -0.8266220 -0.6559880 -0.7465363
```

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

```
## [1]  9.981204 14.162413 10.540220 10.688725 10.689067
```

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

```
## [1] 0.52 0.90 0.70 0.68 0.68
```

```r
Avg_ci_len_k0.8 = colMeans(result_k0.8$confidence_interval_length_matrix)
Avg_ci_len_k0.8
```

```
## [1] 23.03171 59.37230 29.02133 28.91174 28.17429
```

```r
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, Avg_ci_len = Avg_ci_len_k0.8)
```

RESULT

```r
## 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,
                avg_res$Avg_ci_len), ncol=5,nrow = 5)
  vec = round(c(as.numeric(avg_res$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",
                     "CI_length_OR","ATE_IOW","bias_IOW","Abs_bias_IOW",
                     "cov_IOW","CI_length_IOW","ATE_IOW2","bias_IOW2",
                     "Abs_bias_IOW2","cov_IOW2","CI_length_IOW2","ATE_DRE1",
                     "bias_DRE1","Abs_bias_DRE1","cov_DRE1","CI_length_DRE1",
                     "ATE_DRE2","bias_DRE2","Abs_bias_DRE2","cov_DRE2",
                     "CI_length_DRE2")

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

avg_tb
```

```
##          Truth  ATE_OR bias_OR Abs_bias_OR cov_OR CI_length_OR ATE_IOW bias_IOW
## K=0    30.2837 29.5034 -0.7802      2.1989   0.96      10.1724 29.4168  -0.8669
## K=0.05 29.4318 30.1214  0.6897      2.7009   0.90      10.0522 29.5588   0.1270
## K=0.2  30.2283 29.3719 -0.8564      2.3678   0.90      10.8751 30.5231   0.2948
## K=0.5  29.7258 30.2287  0.5028      5.2775   0.56      13.8930 30.8855   1.1597
## K=0.8  29.8026 28.5042 -1.2984      9.9812   0.52      23.0317 29.4755  -0.3271
##        Abs_bias_IOW cov_IOW CI_length_IOW ATE_IOW2 bias_IOW2 Abs_bias_IOW2
## K=0          2.9042    0.98       15.1726  29.6312   -0.6525        2.4348
## K=0.05       3.3857    0.92       15.6562  30.1036    0.6718        2.8708
## K=0.2        3.7180    0.90       18.0592  29.4313   -0.7970        2.7450
```

```
## K=0.5        7.5202    0.74       29.0926  30.0087    0.2829        6.2339
## K=0.8       14.1624    0.90       59.3723  28.9760   -0.8266       10.5402
##        cov_IOW2 CI_length_IOW2 ATE_DRE1 bias_DRE1 Abs_bias_DRE1 cov_DRE1
## K=0        0.94        10.7637  29.6823   -0.6014        2.2653     0.94
## K=0.05     0.88        11.3394  30.0796    0.6478        2.8197     0.86
## K=0.2      0.90        12.1163  29.5215   -0.7069        2.6310     0.92
## K=0.5      0.70        16.3881  30.0555    0.3296        6.1746     0.68
## K=0.8      0.70        29.0213  29.1466   -0.6560       10.6887     0.68
##        CI_length_DRE1 ATE_DRE2 bias_DRE2 Abs_bias_DRE2 cov_DRE2 CI_length_DRE2
## K=0           10.5997  29.6819   -0.6018        2.2677     0.94        10.5821
## K=0.05        11.0741  30.0786    0.6469        2.8149     0.86        11.0484
## K=0.2         12.0478  29.5238   -0.7045        2.6274     0.92        11.9963
## K=0.5         16.3999  30.0526    0.3268        6.1453     0.68        16.1895
## K=0.8         28.9117  29.0561   -0.7465       10.6891     0.68        28.1743
```

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

Table1: TATE + bias + absolute bias

```r
## TATE + bias + absolute bias
avg_tb2 = rbind.data.frame(avg_vec_k0[-c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.05[-c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.2[-c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.5[-c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.8[-c(5,6,10,11,15,16,20,21,25,26)])

colnames(avg_tb2) = c("Truth","ATE_OR","bias_OR","Abs_bias_OR","ATE_IOW",
                      "bias_IOW","Abs_bias_IOW","ATE_IOW2","bias_IOW2",
                      "Abs_bias_IOW2","ATE_DRE1","bias_DRE1",
                      "Abs_bias_DRE1","ATE_DRE2","bias_DRE2","Abs_bias_DRE2")

rownames(avg_tb2) = c("K=0","K=0.05","K=0.2","K=0.5","K=0.8")

avg_tb2
```

```
##          Truth  ATE_OR bias_OR Abs_bias_OR ATE_IOW bias_IOW Abs_bias_IOW
## K=0    30.2837 29.5034 -0.7802      2.1989 29.4168  -0.8669       2.9042
## K=0.05 29.4318 30.1214  0.6897      2.7009 29.5588   0.1270       3.3857
## K=0.2  30.2283 29.3719 -0.8564      2.3678 30.5231   0.2948       3.7180
## K=0.5  29.7258 30.2287  0.5028      5.2775 30.8855   1.1597       7.5202
## K=0.8  29.8026 28.5042 -1.2984      9.9812 29.4755  -0.3271      14.1624
##        ATE_IOW2 bias_IOW2 Abs_bias_IOW2 ATE_DRE1 bias_DRE1 Abs_bias_DRE1
## K=0     29.6312   -0.6525        2.4348  29.6823   -0.6014        2.2653
## K=0.05  30.1036    0.6718        2.8708  30.0796    0.6478        2.8197
## K=0.2   29.4313   -0.7970        2.7450  29.5215   -0.7069        2.6310
## K=0.5   30.0087    0.2829        6.2339  30.0555    0.3296        6.1746
## K=0.8   28.9760   -0.8266       10.5402  29.1466   -0.6560       10.6887
##        ATE_DRE2 bias_DRE2 Abs_bias_DRE2
## K=0     29.6819   -0.6018        2.2677
## K=0.05  30.0786    0.6469        2.8149
## K=0.2   29.5238   -0.7045        2.6274
```

```
## K=0.5   30.0526    0.3268        6.1453
## K=0.8   29.0561   -0.7465       10.6891
```

Table2: CI coverage + CI length

```
## CI coverage + CI length
avg_tb3 = rbind.data.frame(avg_vec_k0[c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.05[c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.2[c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.5[c(5,6,10,11,15,16,20,21,25,26)],
                           avg_vec_k0.8[c(5,6,10,11,15,16,20,21,25,26)])

colnames(avg_tb3) = c("cov_OR","CI_length_OR","cov_IOW","CI_length_IOW",
                      "cov_IOW2","CI_length_IOW2","cov_DRE1",
                      "CI_length_DRE1","cov_DRE2","CI_length_DRE2")

rownames(avg_tb3) = c("K=0","K=0.05","K=0.2","K=0.5","K=0.8")
```

Separate tables

```
ATE_sub = avg_tb %>% select(Truth,ATE_OR,ATE_IOW,ATE_IOW2,ATE_DRE1,ATE_DRE2)
ATE_sub
```

```
##          Truth  ATE_OR ATE_IOW ATE_IOW2 ATE_DRE1 ATE_DRE2
## K=0    30.2837 29.5034 29.4168  29.6312  29.6823  29.6819
## K=0.05 29.4318 30.1214 29.5588  30.1036  30.0796  30.0786
## K=0.2  30.2283 29.3719 30.5231  29.4313  29.5215  29.5238
## K=0.5  29.7258 30.2287 30.8855  30.0087  30.0555  30.0526
## K=0.8  29.8026 28.5042 29.4755  28.9760  29.1466  29.0561
```

```
bias_sub = avg_tb %>% select(bias_OR,bias_IOW,bias_IOW2,bias_DRE1,bias_DRE2)
bias_sub
```

```
##        bias_OR bias_IOW bias_IOW2 bias_DRE1 bias_DRE2
## K=0    -0.7802  -0.8669   -0.6525   -0.6014   -0.6018
## K=0.05  0.6897   0.1270    0.6718    0.6478    0.6469
## K=0.2  -0.8564   0.2948   -0.7970   -0.7069   -0.7045
## K=0.5   0.5028   1.1597    0.2829    0.3296    0.3268
## K=0.8  -1.2984  -0.3271   -0.8266   -0.6560   -0.7465
```

```
Abs_bias_sub = avg_tb %>% select(Abs_bias_OR,Abs_bias_IOW,Abs_bias_IOW2,
                                 Abs_bias_DRE1,Abs_bias_DRE2)
Abs_bias_sub
```

```
##        Abs_bias_OR Abs_bias_IOW Abs_bias_IOW2 Abs_bias_DRE1 Abs_bias_DRE2
## K=0         2.1989       2.9042        2.4348        2.2653        2.2677
## K=0.05      2.7009       3.3857        2.8708        2.8197        2.8149
## K=0.2       2.3678       3.7180        2.7450        2.6310        2.6274
## K=0.5       5.2775       7.5202        6.2339        6.1746        6.1453
## K=0.8       9.9812      14.1624       10.5402       10.6887       10.6891
```

```r
cov_sub = avg_tb %>% select(cov_OR,cov_IOW,cov_IOW2,cov_DRE1,cov_DRE2)
cov_sub
```

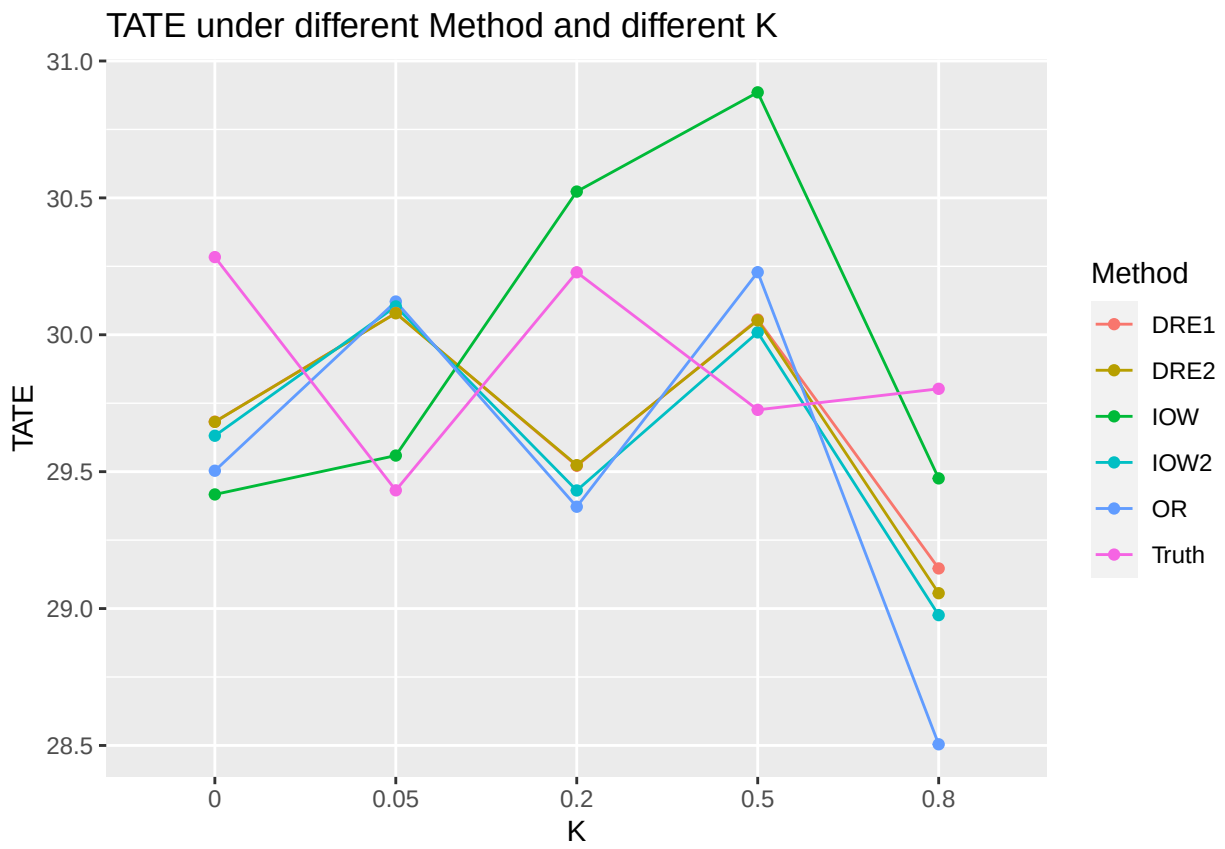
```
##        cov_OR cov_IOW cov_IOW2 cov_DRE1 cov_DRE2
## K=0      0.96    0.98     0.94     0.94     0.94
## K=0.05   0.90    0.92     0.88     0.86     0.86
## K=0.2    0.90    0.90     0.90     0.92     0.92
## K=0.5    0.56    0.74     0.70     0.68     0.68
## K=0.8    0.52    0.90     0.70     0.68     0.68
```

```r
CI_length_sub = avg_tb %>% select(CI_length_OR,CI_length_IOW,CI_length_IOW2,
                                  CI_length_DRE1,CI_length_DRE2)
CI_length_sub
```

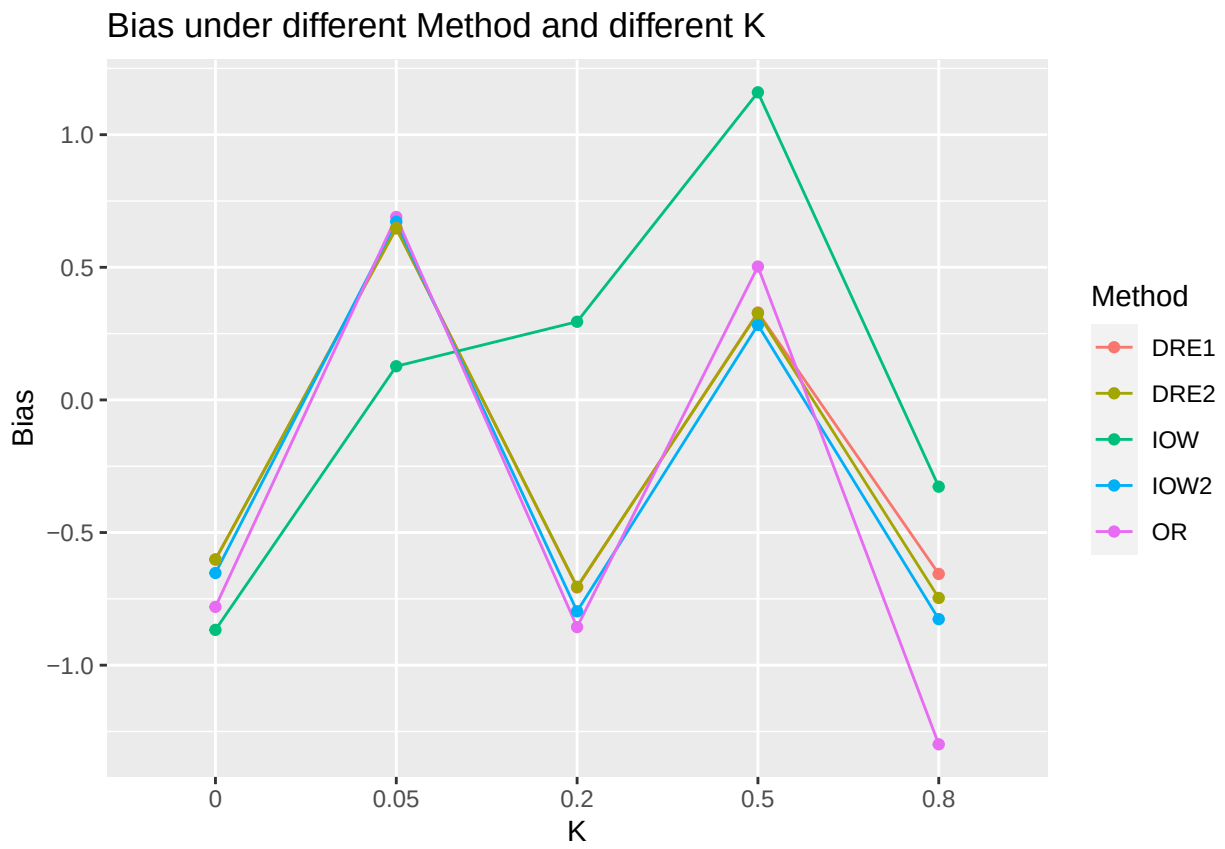
```
##        CI_length_OR CI_length_IOW CI_length_IOW2 CI_length_DRE1 CI_length_DRE2
## K=0         10.1724       15.1726        10.7637        10.5997        10.5821
## K=0.05      10.0522       15.6562        11.3394        11.0741        11.0484
## K=0.2       10.8751       18.0592        12.1163        12.0478        11.9963
## K=0.5       13.8930       29.0926        16.3881        16.3999        16.1895
## K=0.8       23.0317       59.3723        29.0213        28.9117        28.1743
```

Here is for graphing

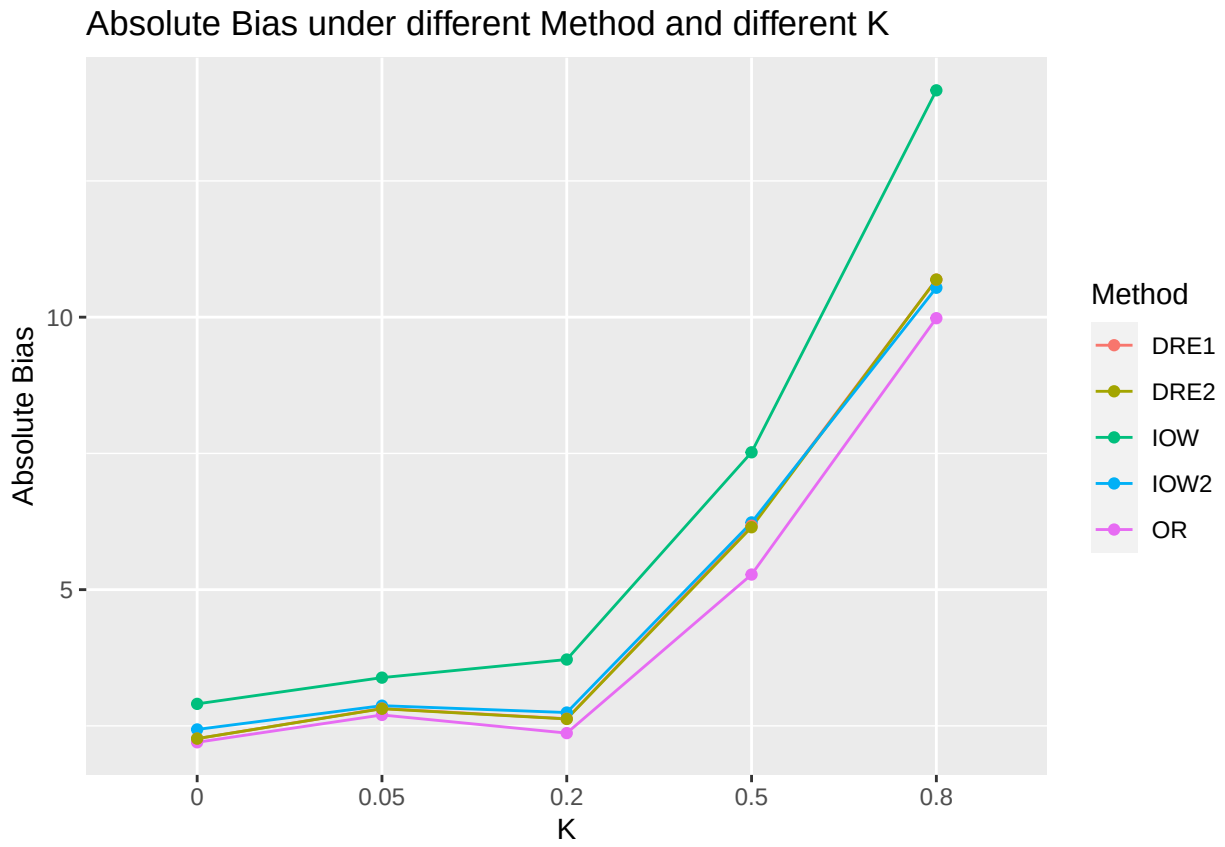
```r
library(ggplot2)
k_lst = c("0","0.05","0.2","0.5","0.8")

##### ATE #####
method = rep(c("Truth","OR" ,"IOW" ,"IOW2" ,"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)

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("TATE") +
  ggtitle("TATE under different Method and different K")
```



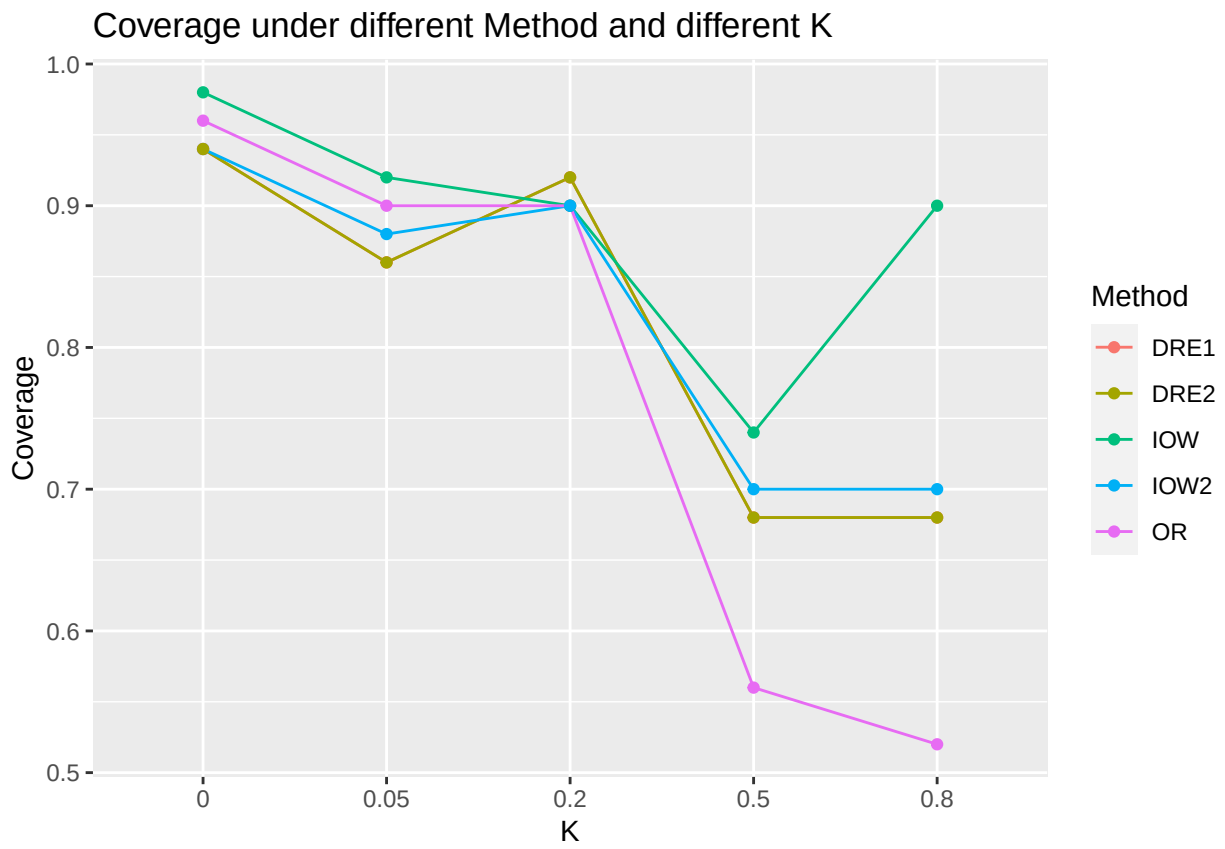

```
##### Bias #####
method = rep(c("OR" ,"IOW" ,"IOW2" ,"DRE1" ,"DRE2"),each = length(k_lst))
k = rep(k_lst,5)
bias_lst = as.numeric(unlist(bias_sub))
bias_tb = cbind.data.frame(Method = method,K = k,bias = bias_lst)

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")
```




```
##### Absolute Bias #####
method = rep(c("OR" ,"IOW" ,"IOW2" ,"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)

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("Absolute Bias") +
  ggtitle("Absolute Bias under different Method and different K")
```




```
##### Coverage #####
method = rep(c("OR" ,"IOW" ,"IOW2" ,"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)

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")
```




```
## CI length
create_long = function(result){
  Method = rep(c("OR" ,"IOW" ,"IOW2" ,"DRE1" ,"DRE2"),
               each = nrow(result$confidence_interval_length_matrix))
  CI_length = c(result$confidence_interval_length_matrix[,1],
             result$confidence_interval_length_matrix[,2],
             result$confidence_interval_length_matrix[,3],
             result$confidence_interval_length_matrix[,4],
             result$confidence_interval_length_matrix[,5])
  long_tb = cbind.data.frame(CI_length,Method)
  long_tb$Method = factor(long_tb$Method,
                          levels = c("OR" ,"IOW" ,"IOW2","DRE1" ,"DRE2"))
  return(long_tb)
}
long_tb_k0 = create_long(result_k0)
long_tb_k0.05 = create_long(result_k0.05)
long_tb_k0.2 = create_long(result_k0.2)
long_tb_k0.5 = create_long(result_k0.5)
long_tb_k0.8 = create_long(result_k0.8)

long_tb_fivek = rbind.data.frame(long_tb_k0,long_tb_k0.05,long_tb_k0.2,
                                 long_tb_k0.5,long_tb_k0.8)
K_long = rep(paste0("k=",k_lst),each= nrow(long_tb_k0))

long_CI_len_all = cbind.data.frame(long_tb_fivek, K = K_long)
```

```
CI_plot <- ggplot(long_CI_len_all, aes(x=Method,y=CI_length,fill=Method))+
  geom_boxplot() +
  labs(title="Confidence Interval Length under different Method and different K") +facet_wrap(~K)+
  ylab("Confidence Interval Length") + theme(legend.position="bottom")
CI_plot
```