matrix fafchamps

```
rm(list = ls())
setwd('/Users/yuezhou/Desktop/Fafchamps/')
library(car)
```

Simulate positions in a two-dimentional space

```
n <- 100
x <- rnorm(n, mean = 0, sd = 1)
y <- rnorm(n, mean= 0, sd = 1)
trials <- 100

#compute the n*n matrix of Euclidean distances between individuals
A <- cbind(x, y)
distance <- dist(A)
distance <- as.matrix(distance)
```

Get the relationship between each pair of individual people

```
#create symmetric matrixs A, B and C as constant that times distance to simulate b
enefit and cost from a relationship
a_0 <- 10
b_0 <- 5
c_0 <- 5

constant <- function(a_0){
  a <- rnorm(n*n, mean = a_0, sd = 1)
  A <- matrix(a, nrow = n)
  for(i in 1:n){
    for(j in 1:n){
      A[j,i] <- A[i,j]
    }
  }
  return(A)
}
A <- constant(a_0)
B <- constant(b_0)
C <- constant(c_0)

B_1 <- A * distance
B_0 <- B * distance
cost <- C * distance
benefit <- B_1 - B_0
relation <- benefit - cost
```

Basic data

```
# number of person
N <- n
# number of variable
K <- 2
# X_ij, i = 1,...,N, j = 1,...,K
X <- matrix(0, N, K)

# L_ij, i = 1,...,N, j = 1,...,N
# simulate each pair of X connect or not, ignore when i = j
#0 means there is no relationship between individuals i and j
#1 means there is a relationship beteen individuals i and j
L <- matrix(0, N, N)
for(i in 1:N){
  for(j in 1:N){
    if(i != j && relation[i, j] >0){
      L[i, j] <- 1
    }
  }
}

beta0 <- 0
beta1 <- 2
beta2 <- 3
beta3 <- 4
beta4 <- 5
beta5 <- 6

confidence_1 <- matrix(NA, nrow = trials, ncol = 4)
confidence_2 <- matrix(NA, nrow = trials, ncol = 4)
confidence_3 <- matrix(NA, nrow = trials, ncol = 4)
confidence_4 <- matrix(NA, nrow = trials, ncol = 4)
confidence_5 <- matrix(NA, nrow = trials, ncol = 4)
```

Now find which rows have common index save as two vectors: left and right

```
# e.g. left = c(a,b,c,d,...)
#      right = c(e,f,g,h,...)
#      --> row (a, e), (b, f), ... share same index
# !! This part is calculated outside of Simulation loop, so only once

whichrow <- function(i, j, N){
  return((i - 1) * (N - 1) + j - as.numeric(j > i))
}

add <- 0
length <- (4 * (N-2) + 2) * (N-1) * N
left <- rep(0, length = length)
right <- rep(0, length = length)
for(i in 1:N){
  cat(i)
```

```
  for(j in 1:N){
    if(i != j){
      # which row in X_long is the (i, j) pair
      leftrow <- whichrow(i, j, N)
      ###################################################################
      # first case (k, l) = (i or j, anything not i, j) -> 2 * (N-1) in total
      # save the added rows to left and right
      # left is the same

      left[1:(2*(N-2)) + add] <- rep(leftrow, 2 * (N - 2))

      # right changes every time
      anything_not_ij <- (1:N)[-c(i, j)]
      right[1:(2*(N-2)) + add] <- c(whichrow(i, anything_not_ij, N),
                                    whichrow(j, anything_not_ij, N))

      ###################################################################
      # second case (k, l) = (anything not i, j, i or j)
      # left_to_add is the same
      left[(2 * (N-2)+1) : (4 * (N-2)) + add] <- rep(leftrow, 2 * (N - 2))
      right[(2 * (N-2)+1) : (4 * (N-2)) + add] <- c(whichrow(anything_not_ij, i,
N),
                                                    whichrow(anything_not_ij, j,
N))

      ###################################################################
      # last case (k, l) = (i, j) or (j, i)
      left[(4 * (N - 2) + 1) : (4 * (N - 2) + 2) + add] <- c(whichrow(i, j, N), wh
ichrow(i, j, N))
      right[(4 * (N - 2) + 1) : (4 * (N - 2) + 2) + add] <- c(whichrow(i, j, N), w
hichrow(j, i, N))
      add <- add + 4 * (N - 2) + 2
    }
  }
}
```

```
## 12345678910111213141516171819202122232425262728293031323334353637383940414243444
4546474849505152535455565758596061626364656667686970717273747576777879808182838485
86878889909192939495969798991OO
```

Calculate variance based on the paper and run many trials

```
for (h in 1 : trials) {
  age1 <- runif(n, min = 0, max =100)
  age2 <- age1
  gender1 <- rbinom(n, 1, 1/2)
  gender2 <- gender1
  X<- cbind (gender1, age1)
  #so that when generating the outcome
  #the outcome with each pair of individual can have a corresponding age, gender a
```

```
nd link
  # Y_ij, i = 1,...,N, j = 1,...,N
  Y <- matrix(0, N, N)
  for(i in 1:N){
    for(j in 1:N){
      if(i != j){
        Y[i, j] <- beta0 + beta1*gender1[i] + beta2*age1[i] + beta3*gender2[j] + b
eta4*age2[j] + beta5*L[i, j]
      }
    }
  }


#################################################################################
  ## regression for only N(N-1) pairs
  ##    Y_long : [Y_11, Y_12, ..., Y_1N, Y_21, Y_22, ..., Y_2N, ..., Y_N(N-1)]
  ##    X_long : [X_11, ..., X_1K, X_21, ..., X_2K, Link_12;
  ##                ...
  ##                X_(N-1)1, ..., X_(N-1)K, X_N1, ..., X_NK, Link_(N-1)N;]
  Y_long <- rep(0, N*(N-1))
  X_long <- matrix(0, N*(N-1), K * 2 + 1)

  count <- 1
  for(i in 1:N){
    for(j in 1:N){
      if(i != j){
        Y_long[count] <- Y[i, j]
        X_long[count, ] <- c(X[i, ], X[j, ], L[i,j])
        count <- count + 1
      }
    }
  }

  ## Y_noise, "someoutcome1"
  Y_noise <-  rnorm(N*(N-1), mean = Y_long, sd = 20)
  ## calculate regular regression, coefficient and standard error
  reg <- lm(Y_noise ~ X_long-1)

  fitted <- as.vector(fitted(reg))
  coef <- summary(reg)$coefficients[, 1:2]
  u <- Y_long - fitted

#################################################################################
  ## calculate new variance
  ## using X_long: (N-1)N times (2K+1)
  ##         u: (N-1)N long
  ## calculate sum inside:
  Xu <- X_long * u
  # now X_ij * u_ij in paper is one row in Xu
  # that row index is (i - 1) * (N - 1) + j - 1 if j > i
  #                     (i - 1) * (N - 1) + j     if j < i, why?
```

```
  #


#################################################################################
  # ted
  sum_inside <- crossprod(Xu[left, ], Xu[right, ])
  sum_inside <- sum_inside
  XX <- solve((t(X_long)) %*% X_long)
  Var <- XX %*% sum_inside %*% XX

  # paper
  #sum_inside <- sum(Xu[left, ] * Xu[right, ])
  #sum_inside <- sum_inside/(2*N * (N*(N-1)-K))
  #XX <- solve((t(X_long)) %*% X_long)
  #Var <- sum_inside * (XX %*% XX)


  Var[Var < 0] <- 0
  # get new SE
  se.new <- diag(Var)^0.5


  #Construct a matrix that contain all the confidence interval generated for each
variable
  conf <- matrix(NA, 5, 4)
  confidence_adjusted <- function(coef, var) {
    for (j in 1 : (2 * K + 1)) {
      conf[j,1] <- coef[j, 1] - 1.96 * var[j,j]^0.5
      conf[j,2] <- coef[j, 1] + 1.96 * var[j,j]^0.5
      conf[j,3] <- coef[j, 1] - 1.96 * coef[j, 2]
      conf[j,4] <- coef[j, 1] + 1.96 * coef[j, 2]
    }

    return(conf)
  }
  confidence <- confidence_adjusted(coef, Var)

  confidence_1[h,] <- confidence[1,]
  confidence_2[h,] <- confidence[2,]
  confidence_3[h,] <- confidence[3,]
  confidence_4[h,] <- confidence[4,]
  confidence_5[h,] <- confidence[5,]

}
```

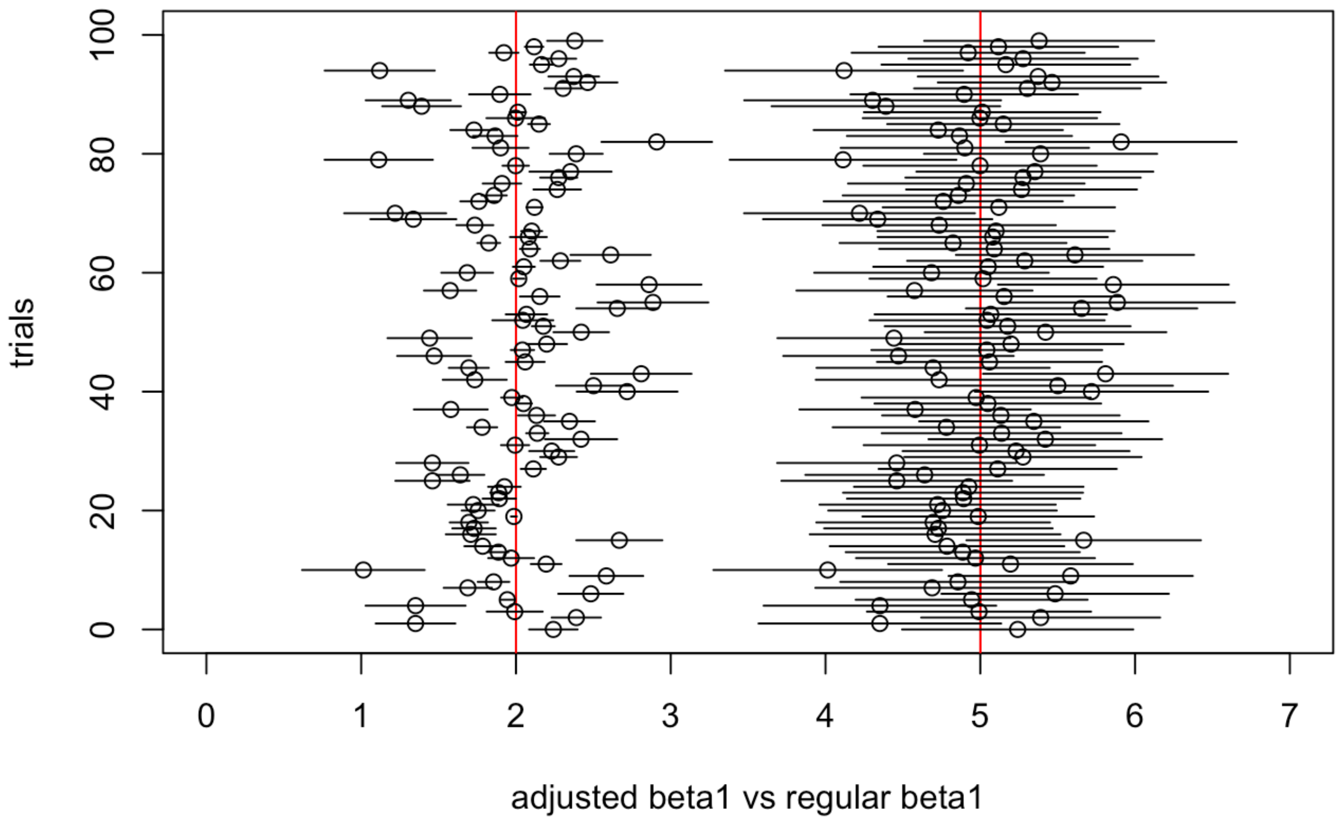
Function that draws the plot for confidence intervals

```
plotint <- function(confidence, beta, x.lab, dis1, dis2, dis3) {
  countbeta <- 0
  countbeta.r <- 0
  dim <- dim(confidence)[1]

  plot(NA, xlim = c(beta - dis1, beta + dis2 + dis3),
       ylim = c(0, dim), xlab = x.lab, ylab = "trials")
  abline(v = beta, col = "red")
  abline(v = beta + dis2, col = "red")
  n_1 <- 0
  for(h in 1:dim){
    segments(x0 = confidence[h,1], y0 = n_1, x1 = confidence[h,2], y1 = n_1)
    segments(x0 = confidence[h,3] + dis2, y0 = n_1, x1 = confidence[h,4] + dis2, y
1 = n_1)
    midpoint_a <- (confidence[h,2] + confidence[h,1]) / 2
    points(midpoint_a, n_1)
    midpoint_r <- (confidence[h,3] + confidence[h,4] + 2 * dis2) / 2
    points(midpoint_r, n_1)
    if (beta < confidence[h,2] && beta > confidence[h,1]) {
      countbeta <- countbeta + 1
    }
    if (beta < confidence[h,4] && beta > confidence[h,3]) {
      countbeta.r <- countbeta.r + 1
    }
    n_1 <- n_1 + 1
  }
  return(c(countbeta, countbeta.r))
}
```

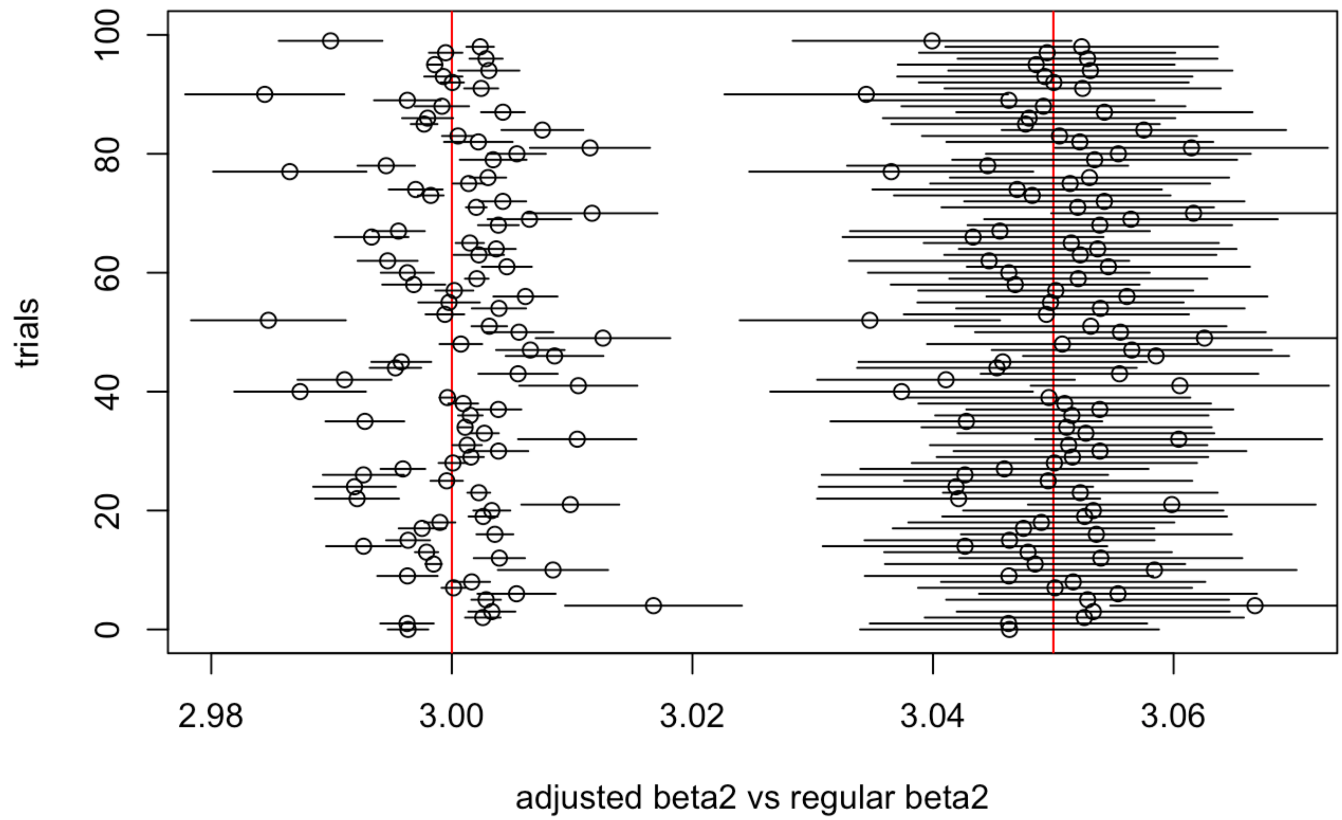
Draw the plots for confidence intervals and calculate the coverage of those two types of confidence interval Red line represents the true value of the variable

```
plotint(confidence_1, beta1, "adjusted beta1 vs regular beta1", 2, 3, 2)
```



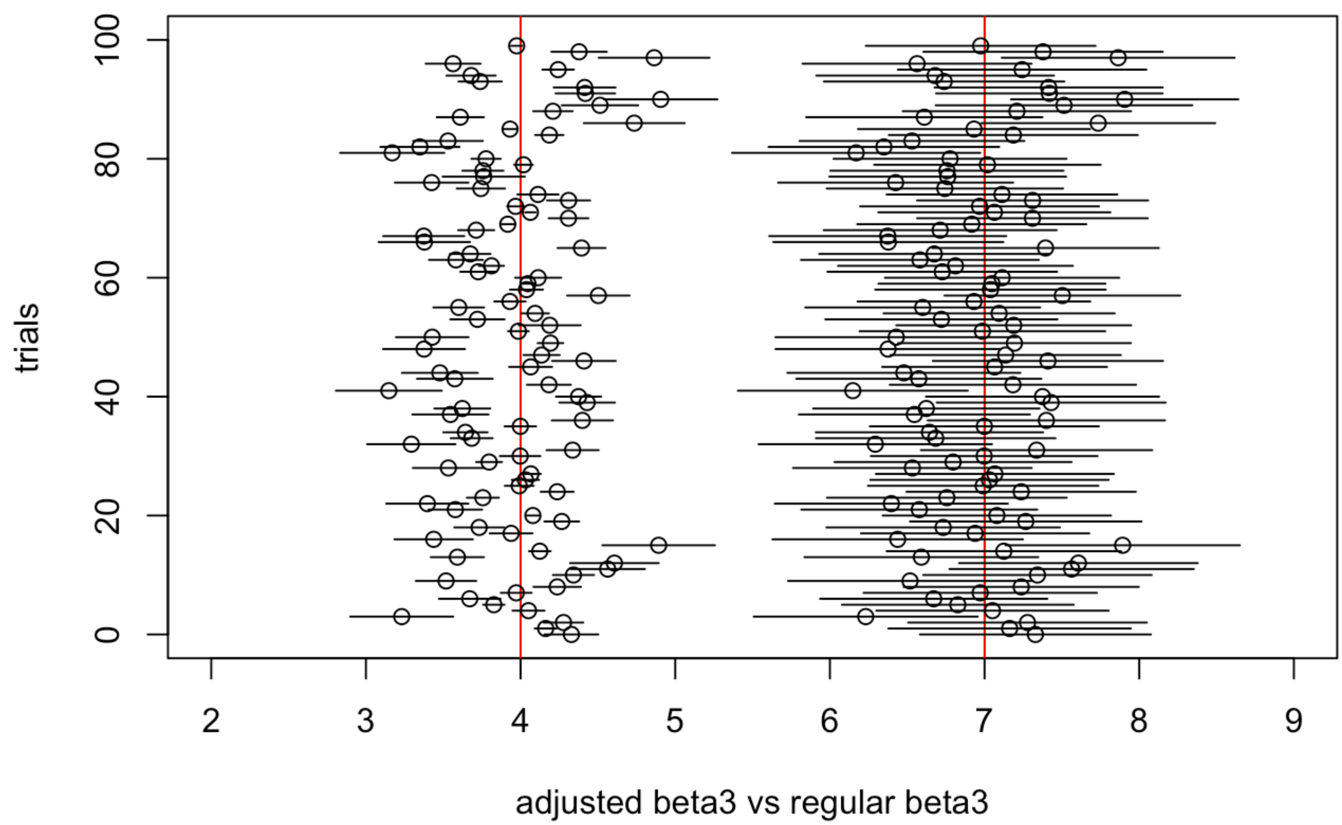

```
## [1] 23 92
```

```
plotint(confidence_2, beta2, "adjusted beta2 vs regular beta2", 0.02, 0.05, 0.02)
```



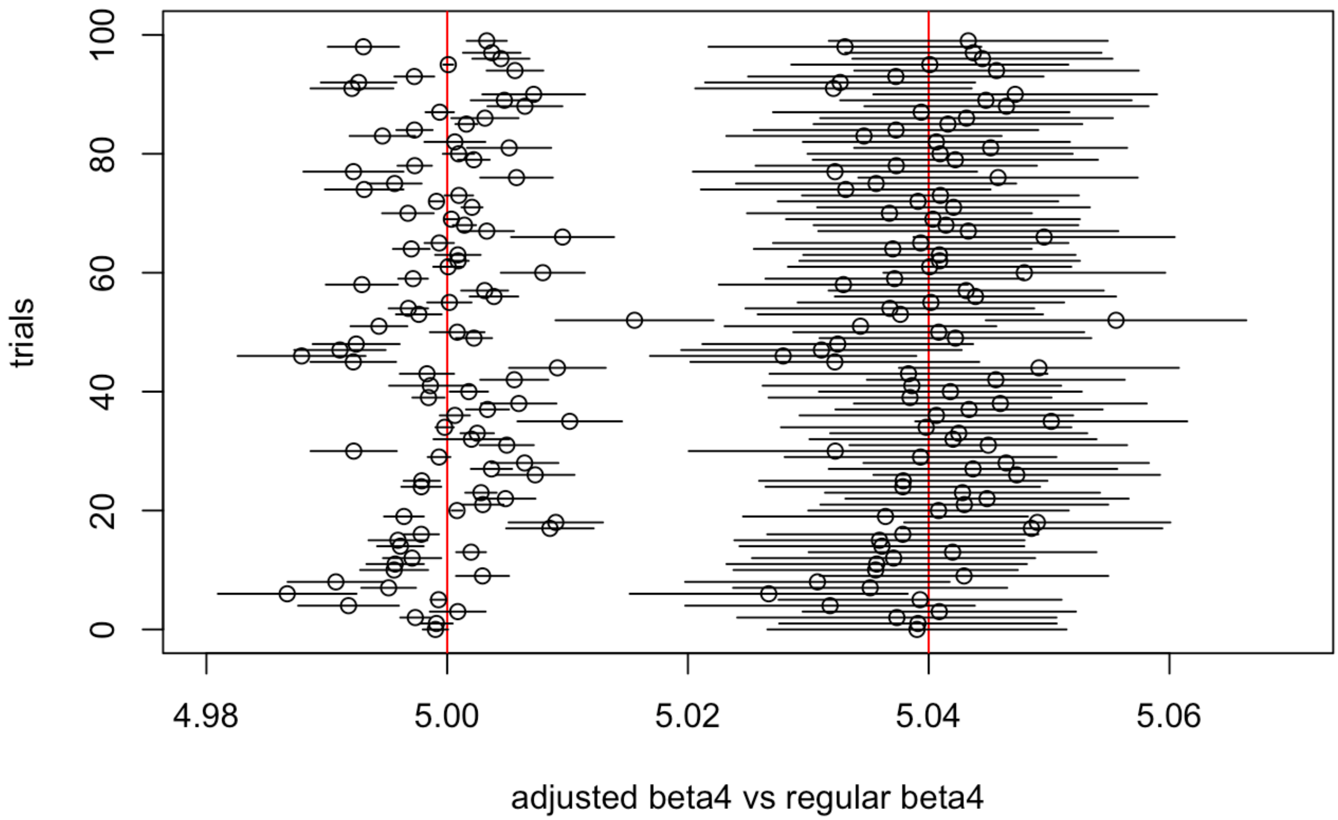

```
## [1] 17 93
```

```
plotint(confidence_3, beta3, "adjusted beta3 vs regular beta3", 2, 3, 2)
```



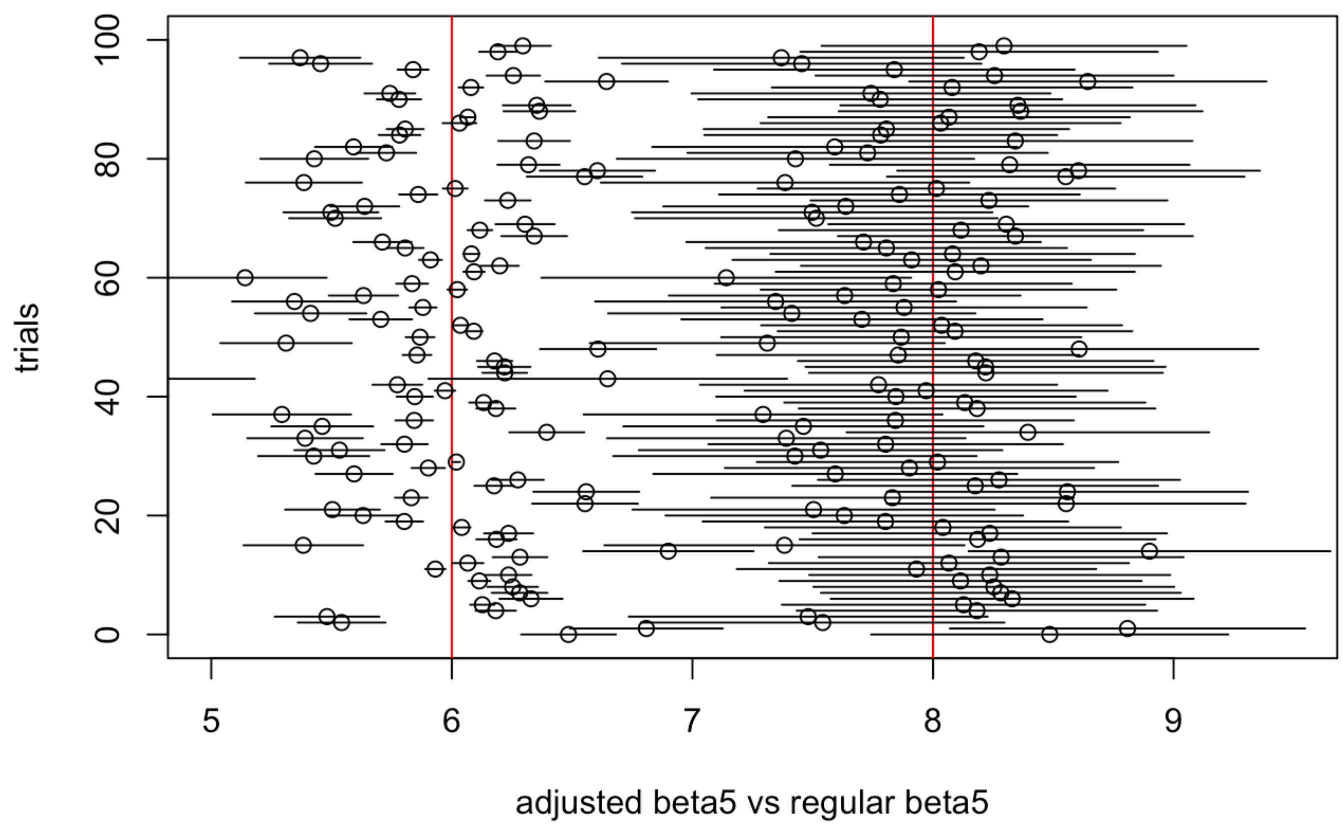

```
## [1] 19 94
```

```
plotint(confidence_4, beta4, "adjusted beta4 vs regular beta4", 0.02, 0.04, 0.03)
```




```
## [1] 20 97
```

```
plotint(confidence_5, beta5, "adjusted beta5 vs regular beta5", 1, 2, 1.5)
```


trials
adjusted beta5 vs regular beta5
0
20
40
60
80
100
5
6
7
8
9

```
## [1]  4 96
```