project2

niyati

2024-11-29

```
logistic_nll <- function(theta, x, y) {
alpha <- theta[1]
beta <- theta[2]
ll <- sum(y * (alpha + beta * x)) - sum(log(1 + exp(alpha + beta *
x)))
return(-ll)
}


confusion <- function(y, yhat){
TN <- sum(y == 0 & yhat == 0)
FP <- sum(y == 0 & yhat == 1)
FN <- sum(y == 1 & yhat == 0)
TP <- sum(y == 1 & yhat == 1)
return(list(TN=TN, FP=FP, FN=FN, TP=TP))
}



### NewtonRaphson functions ###

NewtonRaphson <- function(theta, psiFn, psiPrimeFn, dim, testConvergenceFn = testConvergence,
                          maxIterations = 100, tolerance = 1e-06, relative = FALSE) {
  if (missing(theta)) {
    ## need to figure out the dimensionality
    if (missing(dim)) {
      dim <- length(psiFn())
    }
    theta <- rep(0, dim)
  }
  converged <- FALSE
  i <- 0
  while (!converged & i <= maxIterations) {
    thetaNew <- theta - solve(psiPrimeFn(theta), psiFn(theta))
    converged <- testConvergenceFn(thetaNew, theta, tolerance = tolerance,
                                   relative = relative)

    theta <- thetaNew
    i <- i + 1
  }
  ## Return last value and whether converged or not
  list(theta = theta, converged = converged, iteration = i, fnValue = psiFn(theta))
```

```
}

testConvergence <- function(thetaNew, thetaOld, tolerance = 1e-10, relative = FALSE) {
  sum(abs(thetaNew - thetaOld)) < if (relative)
    tolerance * sum(abs(thetaOld)) else tolerance
}
```

```
### factory functions for Newton-Raphson algorithm  ###

createLogisticPsiFns <- function(x, y) {

  psi <- function(theta = c(0, 0)) {
    alpha <- theta[1]
    beta <- theta[2]
    pu <- exp(alpha + beta * x) / (1 + exp(alpha + beta * x))
    c(sum(y - pu), sum(((y*x) - (pu*x))))
  }

  psiPrime <- function(theta = c(0, 0)) {
    alpha <- theta[1]
    beta <- theta[2]
    pu <- exp(alpha + beta * x) / (1 + exp(alpha + beta * x))^2
    val <- matrix(0, nrow = 2, ncol = 2)
    val[1, 1] <- -sum(pu)
    val[1, 2] <- -sum(pu * x)
    val[2, 1] <- -sum(pu * x)
    val[2, 2] <- -sum(pu * x^2)
    return(val)
  }
  list(psi = psi, psiPrime = psiPrime)
}
```

```
### imported data ###
votes <- read.csv("/Users/niyati/Desktop/votes.csv")

winner <- ifelse(votes$votes_gop_2016 > votes$votes_dem_2016, "Trump", "Clinton")
y = (winner == "Trump")*1
x = votes$Obama
```

```
# algorithm starting values
alpha_0 <- log(mean(y) / (1 - mean(y)))
beta_0 <- 0
# use factory func
psiFns <- createLogisticPsiFns (x = x,y = y)
psi <- psiFns$psi
psiPrime <- psiFns$psiPrime
# call NewtonRaphson
NRresult <- NewtonRaphson(theta = c(alpha_0, beta_0), psiFn = psi, psiPrimeFn = psiPrime)
print(NRresult)
```

```
## $theta
## [1]  20.12942 -37.14827
##
## $converged
## [1] TRUE
```

```
##
## $iteration
## [1] 9
##
## $fnValue
## [1] -3.462231e-13 -1.830758e-13
```

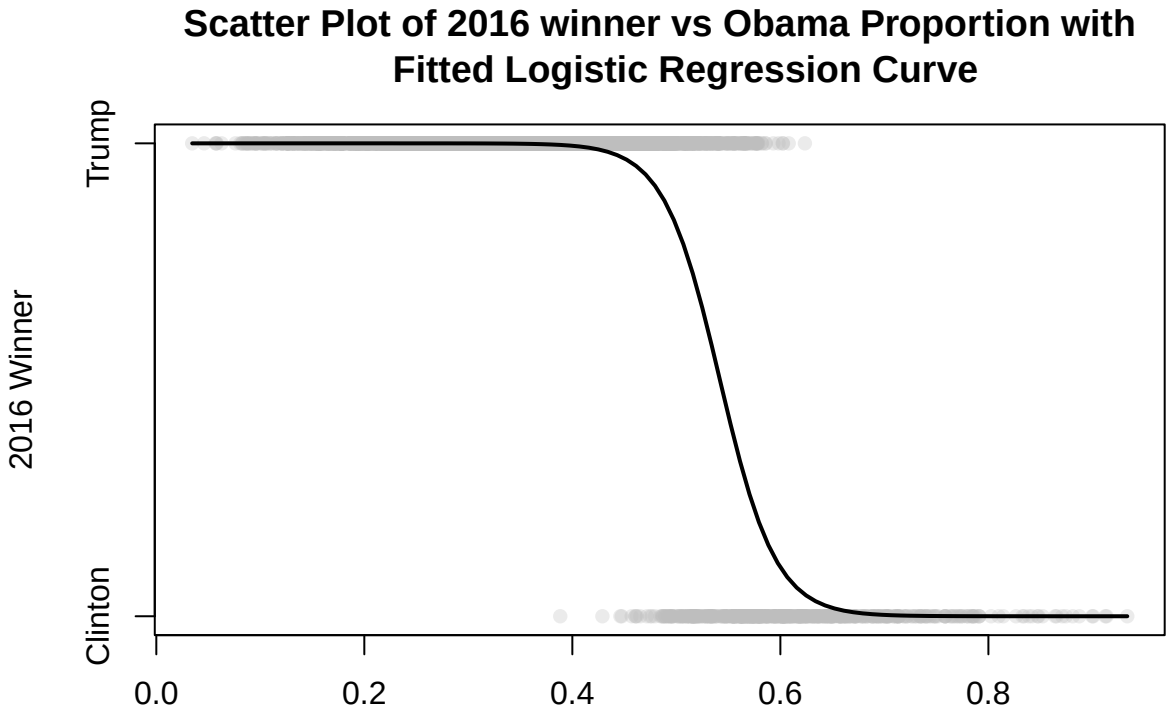
```
alpha_hat <- NRresult$theta[1]
alpha_hat
```

```
## [1] 20.12942
```

```
beta_hat <- NRresult$theta[2]
beta_hat
```

```
## [1] -37.14827
```

```
# Get pu_hat




pu_hat <- exp(alpha_hat + beta_hat * x) / (1 + exp(alpha_hat + beta_hat * x))

plot(x = x, y = y, pch = 16, col = adjustcolor("grey", alpha = 0.3),
     xlab = "The proportion of votes in the county received by Obama in 2012",
     ylab = "2016 Winner", yaxt = 'n',
     main = "Scatter Plot of 2016 winner vs Obama Proportion with
     Fitted Logistic Regression Curve")

axis(2, at = c(0, 1), labels = c("Clinton", "Trump"))

# overlay the fitted logistic regression line
fitted_x <- seq(min(x), max(x), length.out = 100)
fitted_pu <- exp(alpha_hat + beta_hat * fitted_x) /
  (1 + exp(alpha_hat + beta_hat * fitted_x))
lines(fitted_x, fitted_pu, col = "black", lwd = 2)
```



The proportion of votes in the county received by Obama in 2012

```r
# Get pu_hat

pu_hat <- exp(alpha_hat + beta_hat * x) / (1 + exp(alpha_hat + beta_hat * x))
c <- 0.5
yu_hat <- ifelse(pu_hat >= c, 1, 0)
confusion_matrix <- table(y,yu_hat)
dimnames(confusion_matrix) <- list("Truth" = c("Clinton(yu = 0)", "Trump(yu = 1)"),
                                   "Predicted" = c("Clinton(yu_hat = 0)",
                                                   "Trump(yu_hat = 1)"))
print(confusion_matrix)
```

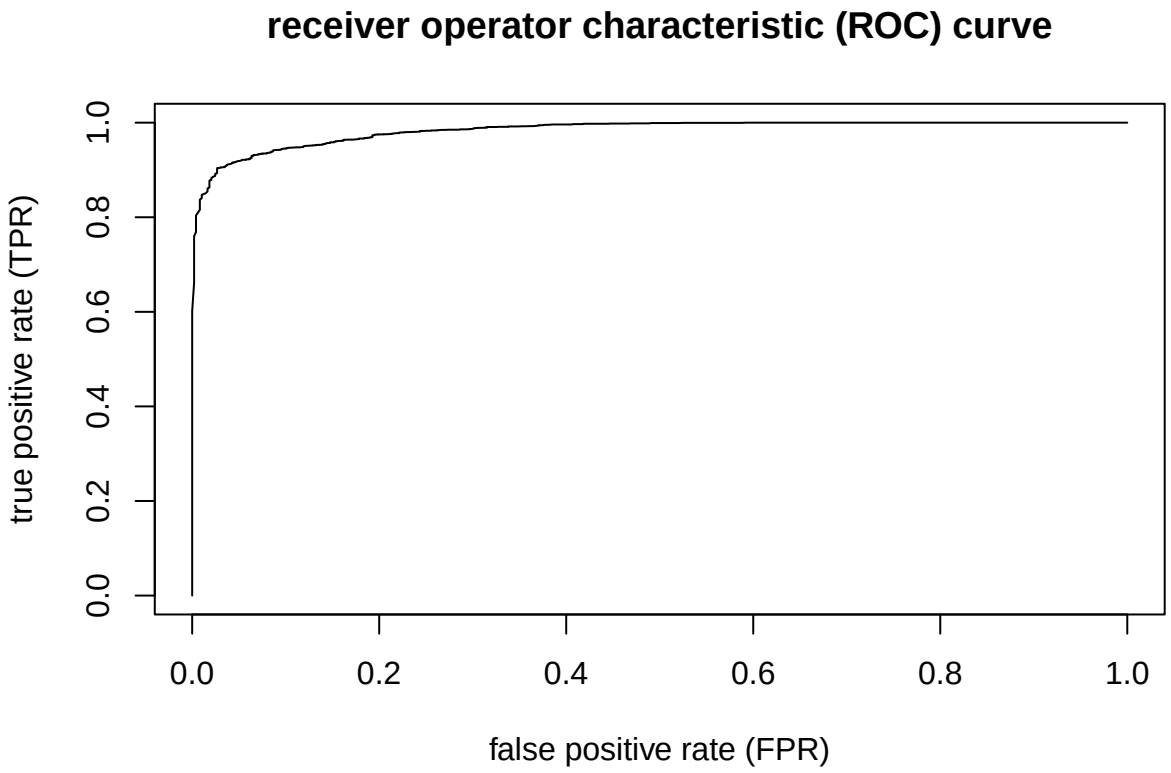
```
##                  Predicted
## Truth             Clinton(yu_hat = 0) Trump(yu_hat = 1)
##   Clinton(yu = 0)                 394                94
##   Trump(yu = 1)                    70              2554
```

```r
# diagonal entries are true positives + true negatives
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(sprintf("Accuracy: %f", accuracy))
```

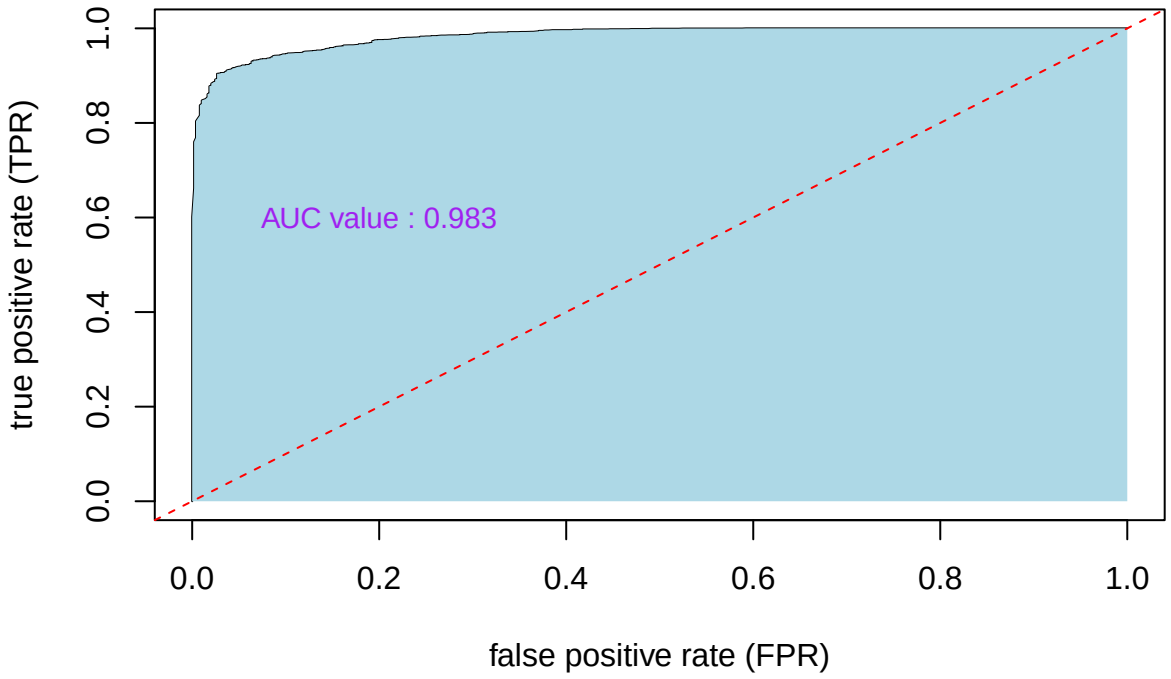
```
## [1] "Accuracy: 0.947301"
```

```r
c_values <- seq(from=0, to=1, length.out=500)
FPR_values <-  rep(0, length(c_values))
TPR_values <-  rep(0, length(c_values))
accuracy_values <- numeric(length(c_values))

for (i in seq_along(c_values)) {
  c <- c_values[i]
  yu_hat <- ifelse(pu_hat >= c, 1, 0)
```

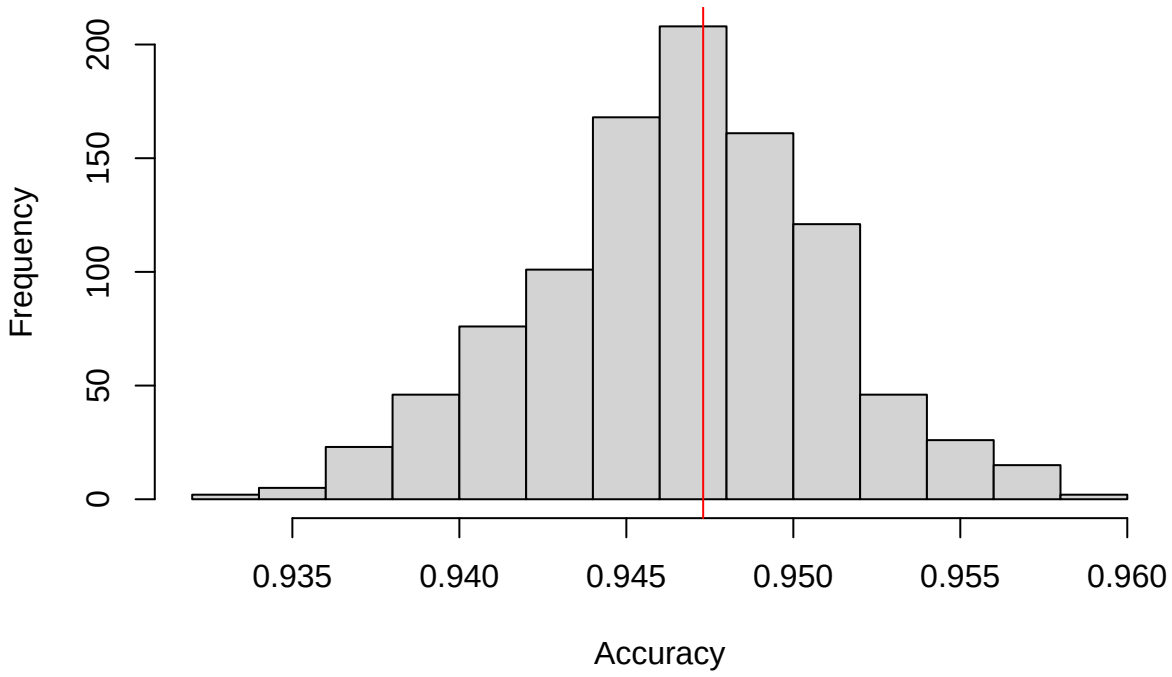
```
  TN <- sum(y == 0 & yu_hat == 0)
  FP <- sum(y == 0 & yu_hat == 1)
  FN <- sum(y == 1 & yu_hat == 0)
  TP <- sum(y == 1 & yu_hat == 1)
  FPR_values[i] <- FP / (FP + TN)
  TPR_values[i] <- TP / (TP + FN)
}

plot(FPR_values, TPR_values,  type = "l",
     ylab = "true positive rate (TPR)", xlab = "false positive rate (FPR)",
     main = "receiver operator characteristic (ROC) curve")
```




```
AUC <- function(x,y){
  d <- data.frame(x,y)
  d <- d[order(d$x),]
  n <- nrow(d)
  sum(diff(d$x) * (d$y[-n]+d$y[-1]))/2
}

plot(FPR_values, TPR_values,  type = "l",
     ylab = "true positive rate (TPR)", xlab = "false positive rate (FPR)",
     main = "receiver operator characteristic (ROC) curve")
# region beneath the curve shaded
polygon(c(FPR_values, 1), c(TPR_values, 0), col = "lightblue", border = FALSE)
auc <- AUC(FPR_values, TPR_values)
text(0.2, 0.6, labels = paste("AUC value :", round(auc, 3)),col = "purple",
     bty = 'n',cex = 0.9)
abline(0, 1, lty = 2, col = "red") #add the line of no discrimination
```



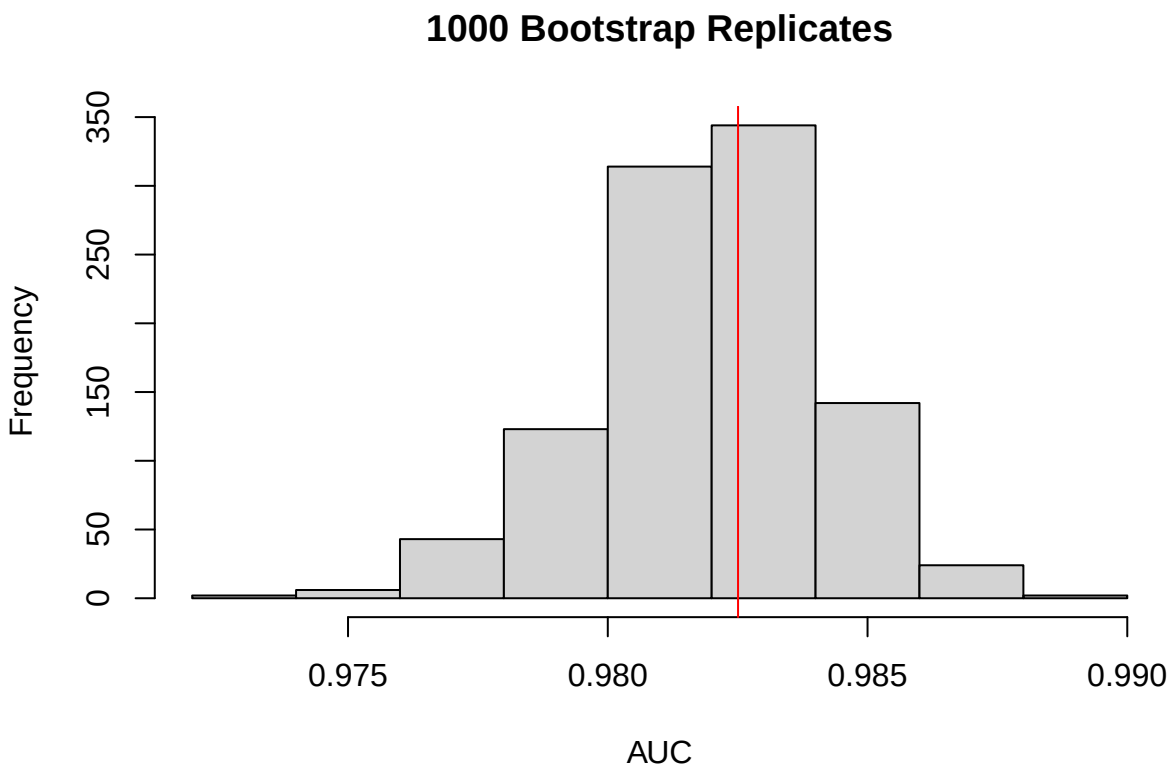

```
popSize <- function(pop) {
  if (is.vector(pop))
  {if (is.logical(pop))
    ## then assume TRUE values identify units
    sum(pop) else length(pop)}
  else nrow(pop)
}
getSample <- function(pop, size, replace=FALSE) {
  N <- popSize(pop)
  pop[sample(1:N, size, replace = replace)]
}
B <- 1000
N <- nrow(votes)
Sstar <- sapply(1:B, FUN = function(b){sample(1:N, N, replace = TRUE)})

accvec <- rep(0, B)
for(b in 1:B){
s <- Sstar[,b]
res <- optim(par = c(0,0), fn = logistic_nll, x=x[s], y=y[s])
alpha_star <- res$par[1]
beta_star <- res$par[2]
phat_star <- exp(alpha_star + beta_star*x[s]) / (1+exp(alpha_star + beta_star*x[s]))
confmat <- confusion(y[s], yhat = (phat_star >= 0.5)*1)
accvec[b] <- (confmat$TP + confmat$TN) / (confmat$TP + confmat$TN + confmat$FP + confmat$FN)
}
hist(accvec, main = "1000 Bootstrap Replicates",
xlab = "Accuracy")
abline(v = accuracy, col = "red")
```




```
### 95% confidence interval for the true population accuracy using the percentile method ###
c(quantile(accvec, 0.025), quantile(accvec, 0.975))
```

```
##      2.5%     97.5%
## 0.9379820 0.9550129
```

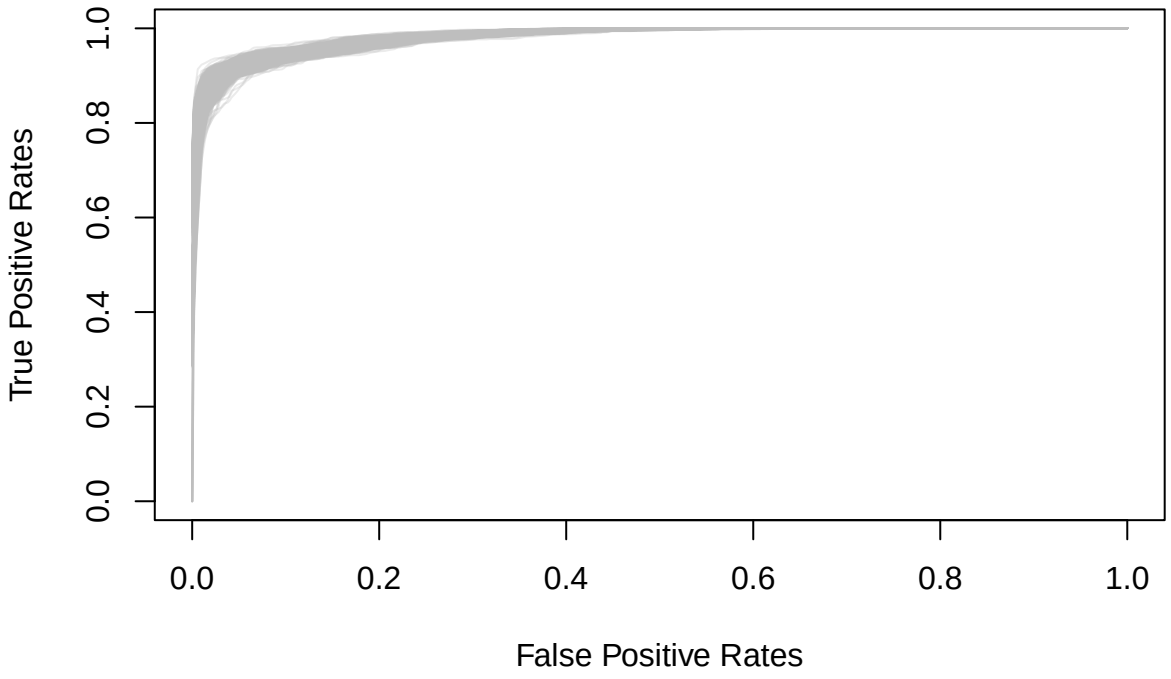
```
aucvec <- rep(0, B)
c <- seq(from = 0, to = 1, length.out = 500)
for(b in 1:B){
s <- Sstar[,b]
res <- optim(par = c(0,0), fn = logistic_nll, x=x[s], y=y[s])
alpha_star <- res$par[1]
beta_star <- res$par[2]
phat_star <- exp(alpha_star + beta_star*x[s]) / (1+exp(alpha_star + beta_star*x[s]))
fpr_star <- rep(0, length(c))
tpr_star <- rep(0, length(c))
for(i in 1:length(c)){
temp <- confusion(y[s], yhat = (phat_star >= c[i])*1)
fpr_star[i] <- temp$FP / (temp$FP + temp$TN)
tpr_star[i] <- temp$TP / (temp$TP + temp$FN)
}
aucvec[b] <- AUC(x = fpr_star, y = tpr_star)
}
hist(aucvec, main = "1000 Bootstrap Replicates",
xlab = "AUC")
abline(v = auc, col = "red")
```



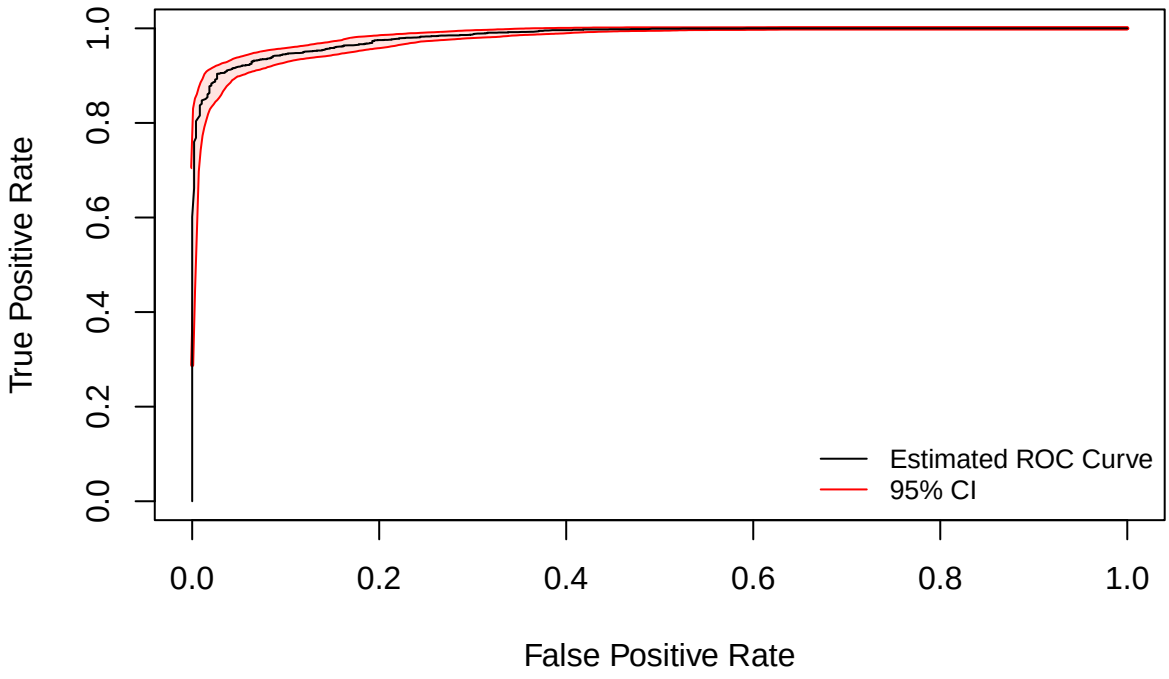

```
quantile(x = aucvec, probs = c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.9772179 0.9860276
```

```
plot(x = 0, y = 0, type = "l", main = "ROC Curves",
xlab = "False Positive Rates", ylim = 0:1, xlim = 0:1,
ylab = "True Positive Rates", col = "white")
c <- seq(from = 0, to = 1, length.out = 500)
roc <- array(data = 0, dim = c(length(c),2,B))
for(b in 1:B){
s <- Sstar[,b]
res <- optim(par = c(0,0), fn = logistic_nll, x=x[s], y=y[s])
alpha_star <- res$par[1]
beta_star <- res$par[2]
phat_star <- exp(alpha_star + beta_star*x[s]) / (1+exp(alpha_star + beta_star*x[s]))
fpr_star <- rep(0, length(c))
tpr_star <- rep(0, length(c))
for(i in 1:length(c)){
temp <- confusion(y[s], yhat = (phat_star >= c[i])*1)
fpr_star[i] <- temp$FP / (temp$FP + temp$TN)
tpr_star[i] <- temp$TP / (temp$TP + temp$FN)
}
temp <- approx(x = fpr_star, y = tpr_star, xout = c, ties = mean)
roc[,1,b] <- temp$x
roc[,2,b] <- temp$y
lines(x = temp$x, y = temp$y, col = adjustcolor("grey", 0.3), lty = 1)
}
```




```
plot(x = 0, y = 0, type = "l", main = "ROC Curve 95% CI",
xlab = "False Positive Rate", ylim = 0:1, xlim = 0:1,
ylab = "True Positive Rate", col = "white")
lo <- apply(X = roc, MARGIN = c(1,2), FUN = quantile, probs = 0.025)
hi <- apply(X = roc, MARGIN = c(1,2), FUN = quantile, probs = 0.975)
lines(lo, col = "red", lwd = 2)
lines(hi, col = "red", lwd = 2)
polygon(x = c(lo[,1],rev(hi[,1])), y = c(lo[,2],rev(hi[,2])),
col = "mistyrose", border = FALSE)
lines(x = FPR_values, y = TPR_values, col = "black")
legend("bottomright", legend = c("Estimated ROC Curve", "95% CI"),
col = c("black", "red"), lty=1, bty = "n", cex=0.8)
```

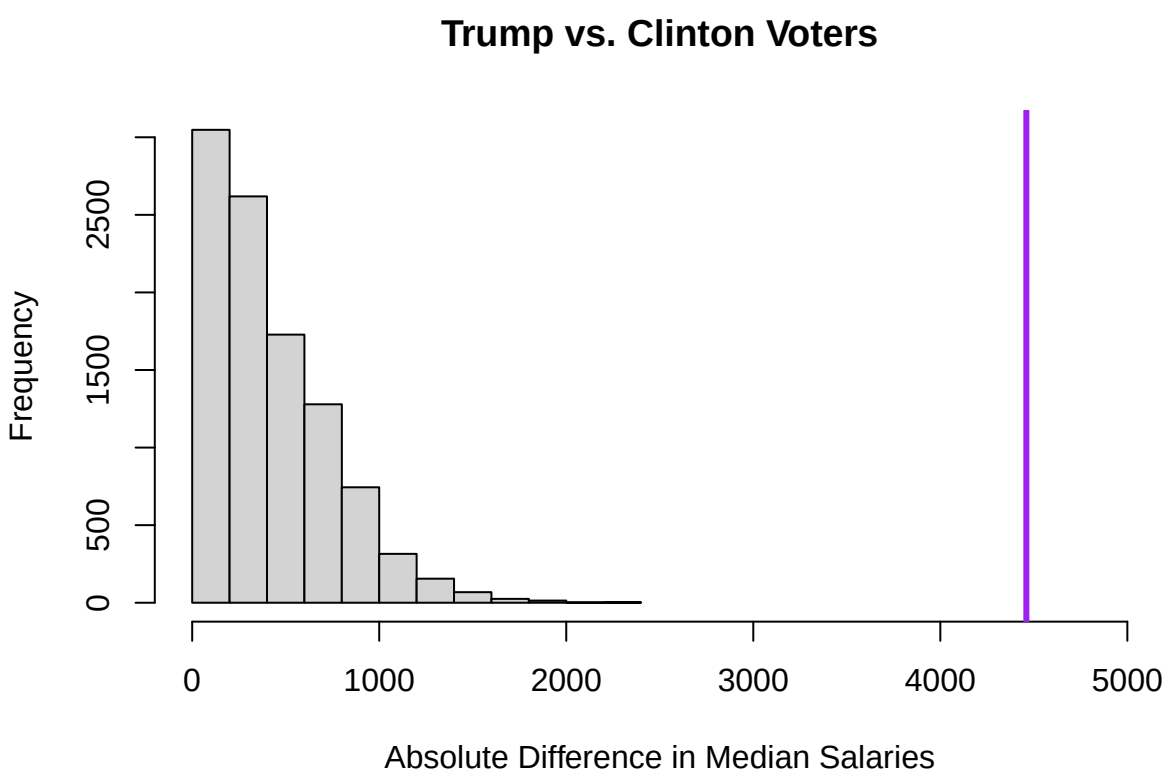



```
### Randomization test ###
## median absolute diff function
mad <- function(y) {
return(median(abs(y - median(y))))
}


mixRandomly <- function(pop){
  pop1 <- pop$pop1
  n_pop1 <- length(pop1)
  pop2 <- pop$pop2
  n_pop2 <- length(pop2)
  mix <- c(pop1,pop2)
  select4pop1 <- sample(1:(n_pop1 + n_pop2), n_pop1, replace = FALSE)
  new_pop1 <- mix[select4pop1]
  new_pop2 <- mix[-select4pop1]
  list(pop1=new_pop1, pop2=new_pop2)
}

#Define PT and PC as sub-populations of INC110213 values for counties won either by Trump or Clinton
# H_null : PT and PC are drawn from the same population of INC110213 values.
# H_null is tht the distributions of median household incomes for Trump-won counties and Clinton-won
# counties are indistinguishable

D1 <- function(pop){
abs(median(pop$pop1) - median(pop$pop2))
}

P <- list(pop1 = votes$INC110213[winner=="Trump"],
pop2 = votes$INC110213[winner=="Clinton"])
```

```r
dobs <- D1(P)
print(dobs)
```

```
## [1] 4460
```

```r
# Randomly mix the populations M = 10, 000 times and construct a histogram of the 10,000
# D1(P_T ,P_C ) values and we find the p-value associated with this test

M <- 10000
D <- rep(0, M)
for(i in 1:M){
Pstar <- mixRandomly(P)
D[i] <- D1(Pstar)
}
hist(D, main = "Trump vs. Clinton Voters", xlim = c(0,5000),
xlab = "Absolute Difference in Median Salaries")
abline(v = dobs, col = "purple", lwd = 3)
```




```r
p_value <- mean(D >= dobs)

p_value
```

```
## [1] 0
```