Lab_Four_Salameh

Sief Salameh

5/20/2023

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse
1.3.1 ──
```

```
## ✓ ggplot2 3.4.2     ✓ purrr   1.0.1
## ✓ tibble  3.2.1     ✓ dplyr   1.1.1
## ✓ tidyr   1.3.0     ✓ stringr 1.5.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
## Warning: package 'tibble' was built under R version 4.1.2
```

```
## Warning: package 'tidyr' was built under R version 4.1.2
```

```
## Warning: package 'readr' was built under R version 4.1.2
```

```
## Warning: package 'purrr' was built under R version 4.1.2
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
## Warning: package 'stringr' was built under R version 4.1.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.2
```

```
setwd("~/Downloads/Machine_Learning/Lab_Four_Salameh")
```

Section One: Exercise 1.1

Part A:

The four kernels supported by svm include:

1) Linear Kernel: The linear kernel is the simplest kernel and is used when the data is linearly separable. It performs a linear transformation, projecting the data points onto a higher-dimensional space. The linear kernel is defined as K(x, y) = x * y.

2) Polynomial Kernel: The polynomial kernel transforms the data into a higher-dimensional space using polynomial functions. It can capture non-linear relationships between the data points. The polynomial kernel is defined as K(x, y) = (α * x * y + c)^d, where α is a scaling factor, c is a constant, and d is the degree of the polynomial.

3) Radial Basis Function (RBF) Kernel: The RBF kernel is a popular choice in SVM because of its flexibility in capturing complex patterns. It transforms the data into an infinite-dimensional space. The RBF is defined as K(x, y) = exp(-γ * ||x - y||^2), where γ is a scaling factor that determines the influence of each training example.

4) Sigmoid Kernel: The sigmoid kernel is another non-linear kernel that is commonly used in SVM. It is inspired by neural networks and can handle non-linear decision boundaries. The sigmoid kernel is defined as K(x, y) = tanh(α * x * y + c), where α is a scaling factor and c is a constant.

CITATION: https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200

Part B:

Typically, the default value of the cost parameter is set to 1.

Part C:

In an SVM classification plot, the X's generally represent the support vectors. Support vectors are usually data points from the training set that lie closest to the decision boundary (hyperplane) of the SVM classifier.

Therefore, support vectors play a critical role in SVM as they determine the position and orientation of the decision boundary. These points have the most influence on defining the boundary and separating the different classes.

When plotting the decision boundary and the support vectors in a graph, the X's tend to be located near the decision boundary and can provide insights into how the SVM model makes its predictions.

Section Two

```
vote_df <- read.csv("vote_df.csv")

work_df <- read.csv("work_df.csv")

apply(vote_df, 2, class)
```

```
##      prtage       pesex    ptdtrace    pehspnon    prcitshp     peeduca
## "character" "character" "character" "character" "character" "character"
##        vote
## "character"
```

```
apply(work_df, 2, class)
```

```
##      prtage       pesex    ptdtrace    pehspnon    prcitshp     peeduca
## "character" "character" "character" "character" "character" "character"
##        work
## "character"
```

```
vote_demo <- vote_df

work_demo <- work_df

vote_demo$prcitshp <- as.factor(vote_demo$prcitshp)

work_demo$prcitshp <- as.factor(work_demo$prcitshp)

str(vote_demo$prcitshp)
```

```
##  Factor w/ 4 levels "FOREIGN BORN, U.S. CITIZEN BY",..: 4 4 1 4 4 4 4 4 4
4 ...
```

```
str(work_demo$prcitshp)
```

```
##  Factor w/ 5 levels "FOREIGN BORN, NOT A CITIZEN OF",..: 5 5 5 5 5 5 5 5 5
5 ...
```

```
prcitshp_unique <- unique(c(vote_df$prcitshp, work_df$prcitshp))

vote_df$prcitshp <- factor(vote_df$prcitshp, levels = prcitshp_unique)

work_df$prcitshp <- factor(work_df$prcitshp, levels = prcitshp_unique)

str(vote_df$prcitshp)
```

```
##  Factor w/ 5 levels "NATIVE, BORN IN THE UNITED",..: 1 1 2 1 1 1 1 1 1 1
...
```

```
str(work_df$prcitshp)
```

```
##  Factor w/ 5 levels "NATIVE, BORN IN THE UNITED",..: 1 1 1 1 1 1 1 1 1 1
...
```

Section Two: Exercise 2.1

```
vote_demo <- vote_df

work_demo <- work_df

# pesex

vote_demo$pesex <- as.factor(vote_demo$pesex)

work_demo$pesex <- as.factor(work_demo$pesex)

str(vote_demo$pesex)
```

```
##  Factor w/ 2 levels "FEMALE","MALE": 1 2 2 2 1 2 1 1 1 2 ...
```

```
str(work_demo$pesex)
```

```
##  Factor w/ 2 levels "FEMALE","MALE": 2 2 2 1 1 2 2 1 2 1 ...
```

```
pesex_unique <- unique(c(vote_df$pesex, work_df$pesex))

vote_df$pesex <- factor(vote_df$pesex, levels = pesex_unique)

work_df$pesex <- factor(work_df$pesex, levels = pesex_unique)

str(vote_df$pesex)
```

```
##  Factor w/ 2 levels "FEMALE","MALE": 1 2 2 2 1 2 1 1 1 2 ...
```

```
str(work_df$pesex)
```

```
##  Factor w/ 2 levels "FEMALE","MALE": 2 2 2 1 1 2 2 1 2 1 ...
```

```
# ptdtrace

vote_demo$ptdtrace <- as.factor(vote_demo$ptdtrace)

work_demo$ptdtrace <- as.factor(work_demo$ptdtrace)

str(vote_demo$ptdtrace)
```

```
##  Factor w/ 15 levels "2 or 3 Races",..: 11 11 11 11 11 11 11 11 11 5 ...
```

```r
str(work_demo$ptdtrace)
##  Factor w/ 13 levels "2 or 3 Races",..: 9 9 9 9 6 9 6 9 9 9 ...
ptdtrace_unique <- unique(c(vote_df$ptdtrace, work_df$ptdtrace))
vote_df$ptdtrace <- factor(vote_df$ptdtrace, levels = ptdtrace_unique)
work_df$ptdtrace <- factor(work_df$ptdtrace, levels = ptdtrace_unique)
str(vote_df$ptdtrace)
##  Factor w/ 16 levels "White Only","Black Only",..: 1 1 1 1 1 1 1 1 1 2 ...
str(work_df$ptdtrace)
##  Factor w/ 16 levels "White Only","Black Only",..: 1 1 1 1 2 1 2 1 1 1 ...
# pehspnon
vote_demo$pehspnon <- as.factor(vote_demo$pehspnon)
work_demo$pehspnon <- as.factor(work_demo$pehspnon)
str(vote_demo$pehspnon)
##  Factor w/ 2 levels "HISPANIC","NON-HISPANIC": 2 2 1 2 2 2 2 2 2 2 ...
str(work_demo$pehspnon)
##  Factor w/ 2 levels "HISPANIC","NON-HISPANIC": 2 2 2 2 2 2 2 2 2 2 ...
pehspnon_unique <- unique(c(vote_df$pehspnon, work_df$pehspnon))
vote_df$pehspnon <- factor(vote_df$pehspnon, levels = pehspnon_unique)
work_df$pehspnon <- factor(work_df$pehspnon, levels = pehspnon_unique)
str(vote_df$pehspnon)
##  Factor w/ 2 levels "NON-HISPANIC",..: 1 1 2 1 1 1 1 1 1 1 ...
str(work_df$pehspnon)
##  Factor w/ 2 levels "NON-HISPANIC",..: 1 1 1 1 1 1 1 1 1 1 ...
# peeduca
vote_demo$peeduca <- as.factor(vote_demo$peeduca)
work_demo$peeduca <- as.factor(work_demo$peeduca)
```

```r
str(vote_demo$peeduca)
```

```
##  Factor w/ 16 levels "10TH GRADE","11TH GRADE",..: 16 14 5 10 16 14 10 10
16 12 ...
```

```r
str(work_demo$peeduca)
```

```
##  Factor w/ 16 levels "10TH GRADE","11TH GRADE",..: 12 12 10 16 16 12 10 16
16 16 ...
```

```r
peeduca_unique <- unique(c(vote_df$peeduca, work_df$peeduca))

vote_df$peeduca <- factor(vote_df$peeduca, levels = peeduca_unique)

work_df$peeduca <- factor(work_df$peeduca, levels = peeduca_unique)

str(vote_df$peeduca)
```

```
##  Factor w/ 16 levels "SOME COLLEGE BUT NO DEGREE",..: 1 2 3 4 1 2 4 4 1 5
...
```

```r
str(work_df$peeduca)
```

```
##  Factor w/ 16 levels "SOME COLLEGE BUT NO DEGREE",..: 5 5 4 1 1 5 4 1 1 1
...
```

```r
# prtage

vote_df$prtage <- as.integer(vote_df$prtage)

work_df$prtage <- as.integer(work_df$prtage)

str(vote_df$prtage)
```

```
##  int [1:5000] 19 35 48 55 25 48 26 23 48 64 ...
```

```r
str(work_df$prtage)
```

```
##  int [1:5000] 35 41 53 21 45 33 34 39 22 25 ...
```

```r
# vote & work

vote_df$vote <- as.factor(vote_df$vote)

work_df$work <- as.factor(work_df$work)

str(vote_df$vote)
```

```
##  Factor w/ 2 levels "did not vote",..: 2 2 2 1 2 1 2 1 1 1 ...
```

```r
str(work_df$work)
```

```
##  Factor w/ 2 levels "flexible","not flexible": 1 1 2 1 1 1 1 1 1 1 ...
```

Section Two: Exercise 2.2

```
cv_svm <- function(k, data, ...) {

  # randomly assign each observation to a fold ---

  initialization <- rep(seq_len(k), nrow(data))

  shuffle <- sample(seq_len(nrow(data)), nrow(data))

  fold_label <- initialization[shuffle]

  # compute the error for each validation set ---

  error <- vector("double", k)

  for (i in seq_len(k)) {
    hold_out <- fold_label == i
    train <- data[!hold_out, ]
    test <- data[hold_out, ]

    # fit the candidate SVM on the training set

    svm_kfold <- svm(work ~ .,
      data = train,
      ...
    )

    predict_kfold <- predict(svm_kfold,
      newdata = test[, !(names(test) %in% c("work"))]
    )

    # compute classification error

    error[i] <- sum(predict_kfold != test[, "work"]) / length(predict_kfold)
  }

  # compute the mean error across the validation sets

  mean(error)
}

cost_values <- c(1, 5, 10)

kernel_values <- c("linear", "sigmoid")
```

```r
models <- expand.grid(cost = cost_values, kernel = kernel_values)

models$error <- NA_real_

for (cost_candidate in cost_values) {
  for (kernel_candidate in kernel_values) {

    # run 5-fold cross-validation for each model

    fold_error <- cv_svm(
      k = 5,
      data = work_df,
      scale = FALSE,
      cost = cost_candidate,
      kernel = kernel_candidate
    )

    # store the cross-validation error in a data frame

    models[models$cost == cost_candidate &
      models$kernel == kernel_candidate, "error"] <- fold_error
  }
}

print(models)
```

```
##   cost  kernel  error
## 1    1  linear 0.1370
## 2    5  linear 0.1390
## 3   10  linear 0.1360
## 4    1 sigmoid 0.3944
## 5    5 sigmoid 0.3944
## 6   10 sigmoid 0.3944
```

```r
(cv_cost <- models[which.min(models$error), "cost"])
```

```
## [1] 10
```

```r
(cv_kernel <- models[which.min(models$error), "kernel"])
```

```
## [1] linear
## Levels: linear sigmoid
```

```r
(cv_error <- models[which.min(models$error), "error"])
```

```
## [1] 0.136
```

In this case, an SVM with a linear kernel and a cost of 10 minimizes the 5-fold cross-validation error rate among the models considered.

Section Two: Exercise 2.3

```
svmfit1 <- svm(work ~ .,
  data = work_df, scale = FALSE,
  cost = 10, kernel = "linear"
)

predict_svmfit <- predict(svmfit1,
  newdata = work_df[, !(names(work_df) %in% c("work"))]
)
(error <- sum(predict_svmfit != work_df[, "work"]) / length(predict_svmfit))

## [1] 0.1338
```

The classification error using the cost value of 10 and the linear kernel model will generate an error rate = .134

Section Two: Exercise 2.4

```
impute_work1 <- predict(svmfit1,
  newdata = vote_df[, !(names(vote_df) %in% c("vote"))]
)

impute_work1 %>% head(5)

##            1            2            3            4            5
##     flexible     flexible not flexible not flexible     flexible
## Levels: flexible not flexible
```

Section Two: Exercise 2.5

```
library(stargazer)

## Warning: package 'stargazer' was built under R version 4.1.2

##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary
Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer

work_numeric <- as.numeric(impute_work1 == "flexible")

vote_numeric <- as.numeric(vote_df$vote == "vote")

reg <- lm(vote_numeric ~ work_numeric + poly(prtage, 2) + pesex,
  data = vote_df
)
```

```r
summary(reg)
```

```
##
## Call:
## lm(formula = vote_numeric ~ work_numeric + poly(prtage, 2) +
##     pesex, data = vote_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.08561 -0.15891  0.01456  0.17932  0.88360
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.34357    0.01133  30.310  < 2e-16 ***
## work_numeric       0.31975    0.01703  18.778  < 2e-16 ***
## poly(prtage, 2)1 -16.31926    0.59673 -27.348  < 2e-16 ***
## poly(prtage, 2)2   1.75413    0.33443   5.245 1.63e-07 ***
## pesexMALE          0.01463    0.00935   1.564    0.118
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3288 on 4995 degrees of freedom
## Multiple R-squared:  0.5669, Adjusted R-squared:  0.5665
## F-statistic:  1635 on 4 and 4995 DF,  p-value: < 2.2e-16
```

```r
work_vote_relationship <- coef(reg)["work_numeric"]

work_vote_relationship
```

```
## work_numeric
##    0.3197453
```

The estimated intercept of the regression line indicates that when all the independent variables are zero, the expected value of "vote_numeric" is 0.34357. This means that the likelihood an individual will vote is 34% when all the independent variables are held constant. The relationship is highly statistically significant.

The estimated coefficient for "work_numeric" is 0.31975. This suggests that for every one-unit increase in "work_numeric," the expected value of "vote_numeric" increases by 0.31975, assuming all the other variables remain constant. In other words, as the work schedule becomes more flexible, the likelihood of voting increases by 32%. This relationship is also highly statistically significant.

The coefficients poly(prtage, 2)1 and poly(prtage, 2)2 correspond to a polynomial transformation of the variable "prtage." The polynomial terms of "prtage" were used to capture nonlinear relationships. Specifically, the coefficient -16.31926 is associated with the linear age, and the coefficient 1.75413 corresponds to the quadratic age squared. The results indicate that if an individual is young, the voting likelihood will decrease by 16% for every year the person ages. However, when the individual is a mature adult or elderly, the

voting likelihood will increase by approximately 2% for every year increase in age. Both relationships are highly statistically significant.

The estimated coefficient for "pesexMALE" is 0.01463. This suggests that, on average, being male (coded as "pesexMALE = 1") is associated with an increase of 0.01463 in the expected value of "vote_numeric" compared to being female (assuming all other variables are held constant). Therefore, being male increases the voting likelihood by 1% compared to being female. This relationship is not statistically significant at any of the confidence levels.

Section Two: Exercise 2.6

```
compute_M <- function(a, b) {
  1 / (1 - 2 * b) * (1 - (1 - b) * b / a - (1 - b) * b / (1 - a))
}

(a <- sum(impute_work1 == "flexible") / length(impute_work1))

## [1] 0.5488

(b <- cv_error)

## [1] 0.136

(M <- compute_M(a, b))

## [1] 0.7217908

work_vote_correction <- work_vote_relationship / M

work_vote_correction

## work_numeric
##    0.4429889
```

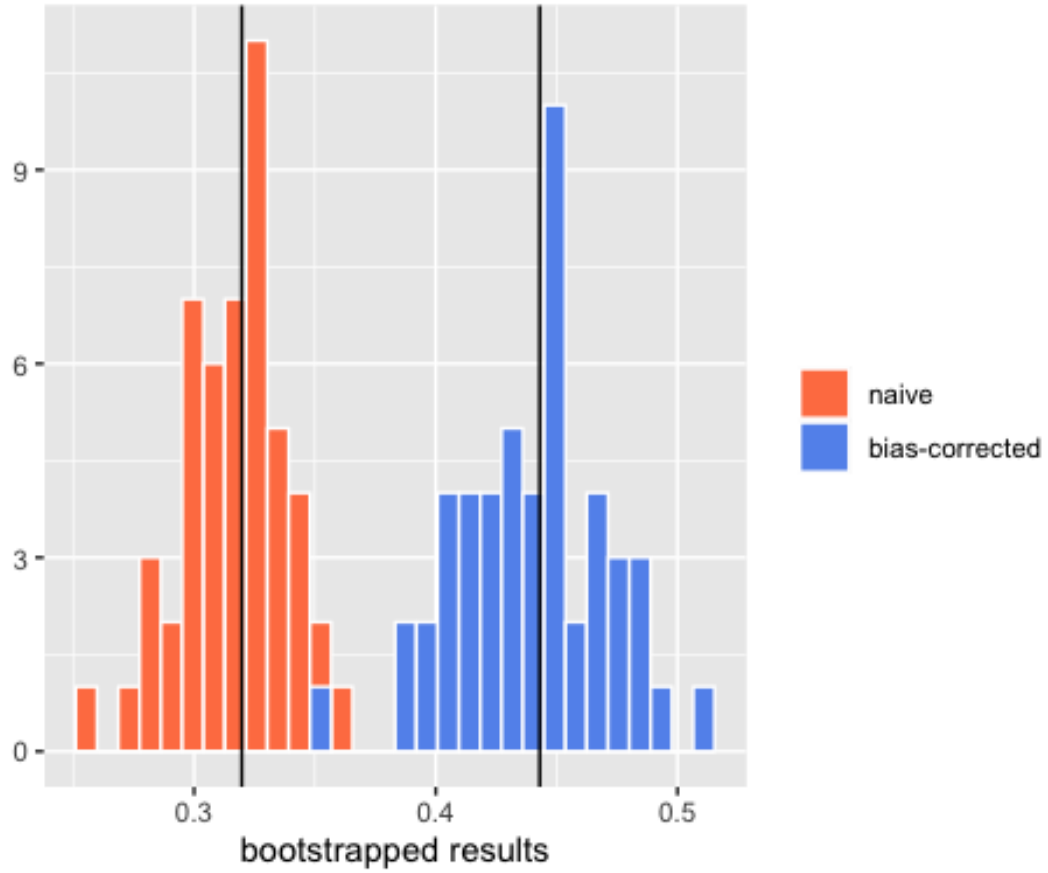
The bias-corrected version is larger. This means that for every unit increase in work flexibility, the likelihood to vote increases by 44%. This indicates a stronger relationship compared to the non-bias-corrected version, which had a 32% likelihood of voting.

Section Two: Exercise 2.7

```
library(ggplot2)

num_bootstrap <- 50

bootstrap_work_vote <- vector("double", num_bootstrap)

bootstrap_work_vote_corrected <- vector("double", num_bootstrap)

for (i in seq_len(num_bootstrap)) {
  bootstrap_index <- sample(nrow(work_df), nrow(work_df), replace = TRUE)
```

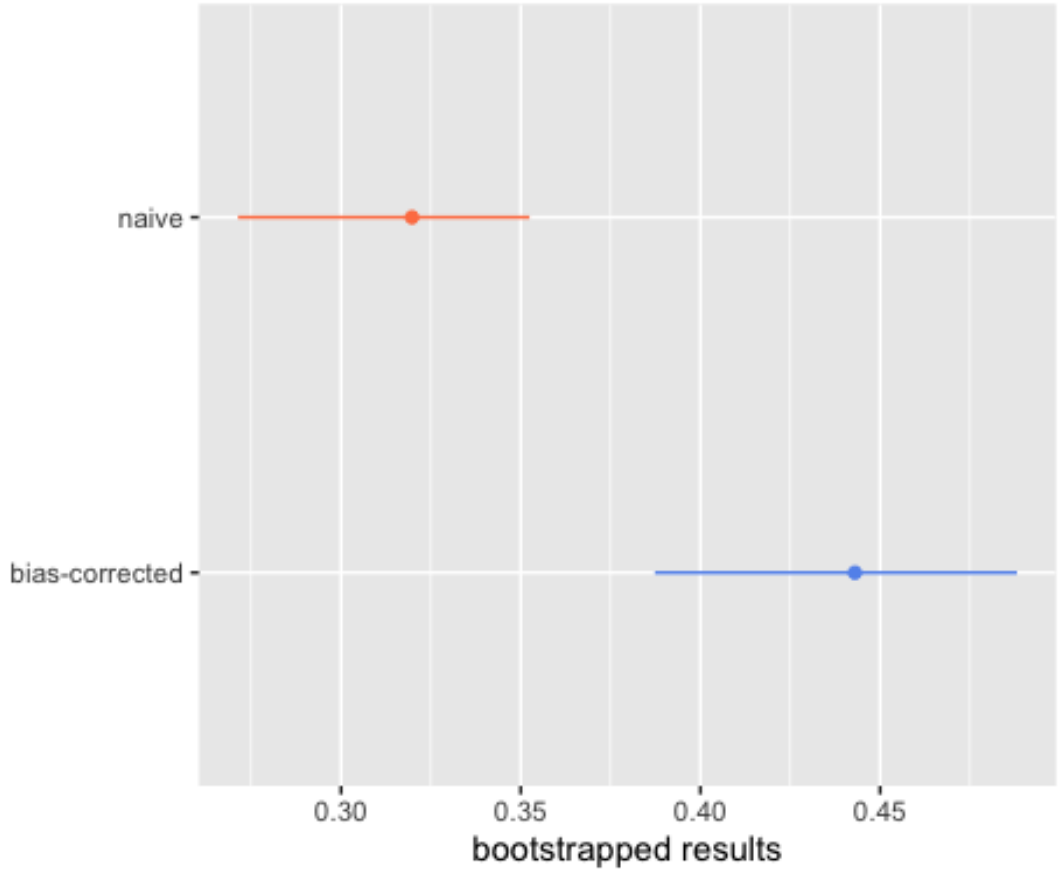
```
  work_df_bootstrap <- work_df[bootstrap_index, ]

  svm_bootstrap <- svm(work ~ .,
    data = work_df_bootstrap,
    scale = FALSE,
    cost = cv_cost,
    kernel = cv_kernel
  )

  impute_bootstrap <- predict(svm_bootstrap,
    newdata = vote_df[!(names(vote_df) %in% c("vote"))]
  )

  a_bootstrap <- sum(impute_bootstrap == "flexible") /
length(impute_bootstrap)

  b_bootstrap <- cv_svm(
    k = 5,
    data = work_df_bootstrap,
    scale = FALSE,
    cost = cv_cost,
    kernel = cv_kernel
  )

  impute_bootstrap_numeric <- as.numeric(impute_bootstrap == "flexible")

  bootstrap_index <- sample(nrow(vote_df), nrow(vote_df), replace = TRUE)

  vote_df_bootstrap <- vote_df[bootstrap_index, ]

  vote_bootstrap <- vote_numeric[bootstrap_index]

  work_bootstrap <- impute_bootstrap_numeric[bootstrap_index]

  reg_bootstrap <- lm(vote_bootstrap ~ work_bootstrap + poly(prtage, 2) +
pesex, data = vote_df_bootstrap)

  bootstrap_work_vote[i] <- coef(reg_bootstrap)["work_bootstrap"]

  M_bootstrap <- compute_M(a_bootstrap, b_bootstrap)

  bootstrap_work_vote_corrected[i] <- bootstrap_work_vote[i] / M_bootstrap
}

ggplot() +
  geom_histogram(aes(x = bootstrap_work_vote, fill = "0"), color = "white") +
  geom_histogram(aes(x = bootstrap_work_vote_corrected, fill = "1"), color =
```

```
"white") +
  scale_fill_manual(
    labels = c("naive", "bias-corrected"),
    values = c("coral", "cornflowerblue"),
    name = ""
  ) +
  geom_vline(xintercept = work_vote_relationship) +
  geom_vline(xintercept = work_vote_correction) +
  xlab("bootstrapped results") +
  ylab("")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Lower/ Upper Bound of 95% confidence interval

naive_ci <- quantile(bootstrap_work_vote, prob = c(0.025, .975))

corrected_ci <- quantile(bootstrap_work_vote_corrected, prob = c(0.025,
.975))

ggplot() +
  geom_segment(aes(
```

```
    x = naive_ci["2.5%"], xend = naive_ci["97.5%"],
    y = "naive", yend = "naive", color = "0"
  )) +
  geom_point(aes(x = work_vote_relationship, y = "naive", color = "0")) +
  geom_segment(aes(
    x = corrected_ci["2.5%"], xend = corrected_ci["97.5%"],
    y = "bias-corrected", yend = "bias-corrected", color = "1"
  )) +
  geom_point(aes(x = work_vote_correction, y = "bias-corrected", color =
"1")) +
  scale_color_manual(
    labels = c("naive", "bias-corrected"),
    values = c("coral", "cornflowerblue")
  ) +
  xlab("bootstrapped results") +
  ylab("") +
  theme(legend.position = "none")
```




```
sd(bootstrap_work_vote)
```

```
## [1] 0.02209112
```

```
sd(bootstrap_work_vote_corrected)
```

```
## [1] 0.03126607
```

Both bootstrap standard error estimates (corrected and not corrected) are larger than the standard error estimate I generated for the work_numeric coefficient in the regression output for exercise 2.5, which was equal to 0.01703.