SampleSplitting

Lei Shi

2024-11-21

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

Setting 1: CRE, splitting by treatment and conditions with no-harm calibration

```
# set up a CRE
set.seed(2025)
MC = 1000
split_setting = c(0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)
# num_models = 2

n1 = 300
n0 = 300
n = n1 + n0

# generate data
X1 = rnorm(n)
X2 = rnorm(n)

Y0 = rpois(n, lambda = exp(0.5 * X1 + 0.5 * X2))
Y1 = rpois(n, lambda = exp(0.3 * X1 + 0.4 * X2))

# true effect
```

```r
tau = mean(Y1 - Y0)

result_correct_model = lapply(1:length(split_setting),
                              function (x){
                                data.frame(
                                  tau = rep(0, MC),
                                  tau_hat_0 = rep(0, MC),
                                  var_hat_0 = rep(0, MC),
                                  tau_hat_cf = rep(0, MC),
                                  var_hat_cf = rep(0, MC)
                                )
                              })

result_wrong_model = lapply(1:length(split_setting),
                            function (x){
                              data.frame(
                                tau = rep(0, MC),
                                tau_hat_0 = rep(0, MC),
                                var_hat_0 = rep(0, MC),
                                tau_hat_cf = rep(0, MC),
                                var_hat_cf = rep(0, MC)
                              )
                            })
```

```r
set.seed(2024)
for (split_index in 1:length(split_setting)){
  print(paste0("split_index: ", split_index))
  pb = txtProgressBar(min = 0, max = MC, initial = 0)
  for (iter in 1:MC){
  setTxtProgressBar(pb, iter)

  # observed data
  Z = sample(c(rep(1, n1), rep(0, n0)))
  Y = Z * Y1 + (1 - Z) * Y0

  # Without calibration
  tau_hat_0 = sum(Y[Z==1])/n1 - sum(Y[Z==0])/n0
  var_hat_0 = var(Y[Z==1])/n1 + var(Y[Z==0])/n0

  # With splitting and calibration
  na = 300
  nb = 300

  na1 = na * split_setting[split_index]
  nb1 = n1 - na1
  na0 = na - na1
  nb0 = nb - nb1

  S1 = sample(c(rep("a", na1), rep("b", nb1)), size = n1)
  S0 = sample(c(rep("a", na0), rep("b", nb0)), size = n0)

  S = rep(NA, n)
  S[Z == 1] = S1
  S[Z == 0] = S0
```

```r
full_data = data.frame(
  Y = Y,
  Z = Z,
  X1 = X1,
  X2 = X2,
  S = S
)

full_data_cf = full_data

# Model fitting with Sample "a" and "b"
glm_fit_a1 = glm(Y~X1+X2, data = full_data%>%filter(S=="a" & Z==1), family = "poisson")
glm_fit_a0 = glm(Y~X1+X2, data = full_data%>%filter(S=="a" & Z==0), family = "poisson")
glm_fit_b1 = glm(Y~X1+X2, data = full_data%>%filter(S=="b" & Z==1), family = "poisson")
glm_fit_b0 = glm(Y~X1+X2, data = full_data%>%filter(S=="b" & Z==0), family = "poisson")

# glm_fit_a1 = glm(Y~X1, data = full_data%>%filter(S=="a" & Z==1), family = "poisson")
# glm_fit_a0 = glm(Y~X1, data = full_data%>%filter(S=="a" & Z==0), family = "poisson")
# glm_fit_b1 = glm(Y~X1, data = full_data%>%filter(S=="b" & Z==1), family = "poisson")
# glm_fit_b0 = glm(Y~X1, data = full_data%>%filter(S=="b" & Z==0), family = "poisson")

pred_a1_of = predict(glm_fit_a1, newdata = full_data%>%filter(S=="a"), type = "response")
pred_a0_of = predict(glm_fit_a0, newdata = full_data%>%filter(S=="a"), type = "response")
pred_b1_of = predict(glm_fit_b1, newdata = full_data%>%filter(S=="b"), type = "response")
pred_b0_of = predict(glm_fit_b0, newdata = full_data%>%filter(S=="b"), type = "response")

pred1 = rep(NA, n)
pred0 = rep(NA, n)
pred1[S=="a"] = pred_a1_of
pred0[S=="a"] = pred_a0_of
pred1[S=="b"] = pred_b1_of
pred0[S=="b"] = pred_b0_of

full_data = cbind(full_data, pred1, pred0)

## Prediction based on cross fitting
pred_a1_cf = predict(glm_fit_b1, newdata = full_data%>%filter(S=="a"), type = "response")
pred_a0_cf = predict(glm_fit_b0, newdata = full_data%>%filter(S=="a"), type = "response")
pred_b1_cf = predict(glm_fit_a1, newdata = full_data%>%filter(S=="b"), type = "response")
pred_b0_cf = predict(glm_fit_a0, newdata = full_data%>%filter(S=="b"), type = "response")
pred1 = rep(NA, n)
pred0 = rep(NA, n)
pred1[S=="a"] = pred_a1_cf
pred0[S=="a"] = pred_a0_cf
pred1[S=="b"] = pred_b1_cf
pred0[S=="b"] = pred_b0_cf

full_data_cf = cbind(full_data_cf, pred1, pred0)

# Linear calibration for the model
lm_fit_a1 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "a" & Z == 1))
lm_fit_a0 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "a" & Z == 0))
lm_fit_b1 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "b" & Z == 1))
```

```
  lm_fit_b0 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "b" & Z == 0))

  # Cross fit based on linear calibration
  pred_a1_cf_lc = predict(lm_fit_b1, newdata = full_data_cf%>%filter(S == "a"))
  pred_a0_cf_lc = predict(lm_fit_b0, newdata = full_data_cf%>%filter(S == "a"))
  pred_b1_cf_lc = predict(lm_fit_a1, newdata = full_data_cf%>%filter(S == "b"))
  pred_b0_cf_lc = predict(lm_fit_a0, newdata = full_data_cf%>%filter(S == "b"))

  pred1_lc = rep(NA, n)
  pred0_lc = rep(NA, n)
  pred1_lc[S == "a"] = pred_a1_cf_lc
  pred0_lc[S == "a"] = pred_a0_cf_lc
  pred1_lc[S == "b"] = pred_b1_cf_lc
  pred0_lc[S == "b"] = pred_b0_cf_lc

  full_data_cf = cbind(full_data_cf, pred1_lc, pred0_lc)
  ## getting residuals
  full_data_cf = full_data_cf %>%
    mutate(epsilon = case_when(
      Z == 1 ~ Y - pred1_lc,
      Z == 0 ~ Y - pred0_lc
    ))

  # Construct estimator
  ## point estimator
  mu_cf_a1 = 1/na * sum(full_data_cf$epsilon[S=="a" & Z==1])/(na1/na) + 1/na * sum(pred1_lc[S=="a"])
  mu_cf_a0 = 1/na * sum(full_data_cf$epsilon[S=="a" & Z==0])/(na0/na) + 1/na * sum(pred0_lc[S=="a"])
  mu_cf_b1 = 1/nb * sum(full_data_cf$epsilon[S=="b" & Z==1])/(nb1/nb) + 1/nb * sum(pred1_lc[S=="b"])
  mu_cf_b0 = 1/nb * sum(full_data_cf$epsilon[S=="b" & Z==0])/(nb0/nb) + 1/nb * sum(pred0_lc[S=="b"])

  tau_hat_cf = na/n * (mu_cf_a1-mu_cf_a0) + nb/n * (mu_cf_b1-mu_cf_b0)
  var_hat_cf =
    na^2/n^2 * (var(full_data_cf$epsilon[S=="a" & Z==1])/na1 + var(full_data_cf$epsilon[S=="a" & Z==0])
    nb^2/n^2 * (var(full_data_cf$epsilon[S=="b" & Z==1])/nb1 + var(full_data_cf$epsilon[S=="b" & Z==0])

  result_correct_model[[split_index]]$tau[iter] = tau
  result_correct_model[[split_index]]$tau_hat_0[iter] = tau_hat_0
  result_correct_model[[split_index]]$var_hat_0[iter] = var_hat_0
  result_correct_model[[split_index]]$tau_hat_cf[iter] = tau_hat_cf
  result_correct_model[[split_index]]$var_hat_cf[iter] = var_hat_cf
}

close(pb)
}
```

```
## [1] "split_index: 1"
## ================================================================================
## [1] "split_index: 2"
## ================================================================================
## [1] "split_index: 3"
## ================================================================================
## [1] "split_index: 4"
## ================================================================================
## [1] "split_index: 5"
```

```
## ================================================================================
## [1] "split_index: 6"
## ================================================================================
## [1] "split_index: 7"
## ================================================================================
```

```r
set.seed(2024)
for (split_index in 1:length(split_setting)){
  print(paste0("split_index: ", split_index))
  pb = txtProgressBar(min = 0, max = MC, initial = 0)
  for (iter in 1:MC){
  setTxtProgressBar(pb, iter)

  # observed data
  Z = sample(c(rep(1, n1), rep(0, n0)))
  Y = Z * Y1 + (1 - Z) * Y0

  # Without calibration
  tau_hat_0 = sum(Y[Z==1])/n1 - sum(Y[Z==0])/n0
  var_hat_0 = var(Y[Z==1])/n1 + var(Y[Z==0])/n0

  # With splitting and calibration
  na = 300
  nb = 300

  na1 = na * split_setting[split_index]
  nb1 = n1 - na1
  na0 = na - na1
  nb0 = nb - nb1

  S1 = sample(c(rep("a", na1), rep("b", nb1)), size = n1)
  S0 = sample(c(rep("a", na0), rep("b", nb0)), size = n0)

  S = rep(NA, n)
  S[Z == 1] = S1
  S[Z == 0] = S0

  full_data = data.frame(
    Y = Y,
    Z = Z,
    X1 = X1,
    X2 = X2,
    S = S
  )

  full_data_cf = full_data

  # Model fitting with Sample "a" and "b"
  # glm_fit_a1 = glm(Y~X1+X2, data = full_data%>%filter(S=="a" & Z==1), family = "poisson")
  # glm_fit_a0 = glm(Y~X1+X2, data = full_data%>%filter(S=="a" & Z==0), family = "poisson")
  # glm_fit_b1 = glm(Y~X1+X2, data = full_data%>%filter(S=="b" & Z==1), family = "poisson")
  # glm_fit_b0 = glm(Y~X1+X2, data = full_data%>%filter(S=="b" & Z==0), family = "poisson")

  glm_fit_a1 = glm(Y~X1, data = full_data%>%filter(S=="a" & Z==1), family = "poisson")
```

```r
glm_fit_a0 = glm(Y~X1, data = full_data%>%filter(S=="a" & Z==0), family = "poisson")
glm_fit_b1 = glm(Y~X1, data = full_data%>%filter(S=="b" & Z==1), family = "poisson")
glm_fit_b0 = glm(Y~X1, data = full_data%>%filter(S=="b" & Z==0), family = "poisson")

pred_a1_of = predict(glm_fit_a1, newdata = full_data%>%filter(S=="a"), type = "response")
pred_a0_of = predict(glm_fit_a0, newdata = full_data%>%filter(S=="a"), type = "response")
pred_b1_of = predict(glm_fit_b1, newdata = full_data%>%filter(S=="b"), type = "response")
pred_b0_of = predict(glm_fit_b0, newdata = full_data%>%filter(S=="b"), type = "response")

pred1 = rep(NA, n)
pred0 = rep(NA, n)
pred1[S=="a"] = pred_a1_of
pred0[S=="a"] = pred_a0_of
pred1[S=="b"] = pred_b1_of
pred0[S=="b"] = pred_b0_of

full_data = cbind(full_data, pred1, pred0)

## Prediction based on cross fitting
pred_a1_cf = predict(glm_fit_b1, newdata = full_data%>%filter(S=="a"), type = "response")
pred_a0_cf = predict(glm_fit_b0, newdata = full_data%>%filter(S=="a"), type = "response")
pred_b1_cf = predict(glm_fit_a1, newdata = full_data%>%filter(S=="b"), type = "response")
pred_b0_cf = predict(glm_fit_a0, newdata = full_data%>%filter(S=="b"), type = "response")
pred1 = rep(NA, n)
pred0 = rep(NA, n)
pred1[S=="a"] = pred_a1_cf
pred0[S=="a"] = pred_a0_cf
pred1[S=="b"] = pred_b1_cf
pred0[S=="b"] = pred_b0_cf

full_data_cf = cbind(full_data_cf, pred1, pred0)

# Linear calibration for the model
lm_fit_a1 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "a" & Z == 1))
lm_fit_a0 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "a" & Z == 0))
lm_fit_b1 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "b" & Z == 1))
lm_fit_b0 = lm(Y~pred1+pred0, data = full_data %>% filter(S == "b" & Z == 0))

# Cross fit based on linear calibration
pred_a1_cf_lc = predict(lm_fit_b1, newdata = full_data_cf%>%filter(S == "a"))
pred_a0_cf_lc = predict(lm_fit_b0, newdata = full_data_cf%>%filter(S == "a"))
pred_b1_cf_lc = predict(lm_fit_a1, newdata = full_data_cf%>%filter(S == "b"))
pred_b0_cf_lc = predict(lm_fit_a0, newdata = full_data_cf%>%filter(S == "b"))

pred1_lc = rep(NA, n)
pred0_lc = rep(NA, n)
pred1_lc[S == "a"] = pred_a1_cf_lc
pred0_lc[S == "a"] = pred_a0_cf_lc
pred1_lc[S == "b"] = pred_b1_cf_lc
pred0_lc[S == "b"] = pred_b0_cf_lc

full_data_cf = cbind(full_data_cf, pred1_lc, pred0_lc)
## getting residuals
```

```
  full_data_cf = full_data_cf %>%
    mutate(epsilon = case_when(
      Z == 1 ~ Y - pred1_lc,
      Z == 0 ~ Y - pred0_lc
    ))

  # Construct estimator
  ## point estimator
  mu_cf_a1 = 1/na * sum(full_data_cf$epsilon[S=="a" & Z==1])/(na1/na) + 1/na * sum(pred1_lc[S=="a"])
  mu_cf_a0 = 1/na * sum(full_data_cf$epsilon[S=="a" & Z==0])/(na0/na) + 1/na * sum(pred0_lc[S=="a"])
  mu_cf_b1 = 1/nb * sum(full_data_cf$epsilon[S=="b" & Z==1])/(nb1/nb) + 1/nb * sum(pred1_lc[S=="b"])
  mu_cf_b0 = 1/nb * sum(full_data_cf$epsilon[S=="b" & Z==0])/(nb0/nb) + 1/nb * sum(pred0_lc[S=="b"])

  tau_hat_cf = na/n * (mu_cf_a1-mu_cf_a0) + nb/n * (mu_cf_b1-mu_cf_b0)
  var_hat_cf =
    na^2/n^2 * (var(full_data_cf$epsilon[S=="a" & Z==1])/na1 + var(full_data_cf$epsilon[S=="a" & Z==0])
    nb^2/n^2 * (var(full_data_cf$epsilon[S=="b" & Z==1])/nb1 + var(full_data_cf$epsilon[S=="b" & Z==0])

  result_wrong_model[[split_index]]$tau[iter] = tau
  result_wrong_model[[split_index]]$tau_hat_0[iter] = tau_hat_0
  result_wrong_model[[split_index]]$var_hat_0[iter] = var_hat_0
  result_wrong_model[[split_index]]$tau_hat_cf[iter] = tau_hat_cf
  result_wrong_model[[split_index]]$var_hat_cf[iter] = var_hat_cf
}

close(pb)
}
```

```
## [1] "split_index: 1"
## ================================================================================
## [1] "split_index: 2"
## ================================================================================
## [1] "split_index: 3"
## ================================================================================
## [1] "split_index: 4"
## ================================================================================
## [1] "split_index: 5"
## ================================================================================
## [1] "split_index: 6"
## ================================================================================
## [1] "split_index: 7"
## ================================================================================
```

Summarize results

combined results

```
# combined data
combined_results_correct_model = do.call(rbind, lapply(1:length(split_setting), function(i) {
  # Add a "setting" column to each data frame
  result_correct_model[[i]]$setting = split_setting[i]
  return(result_correct_model[[i]])
}))
```

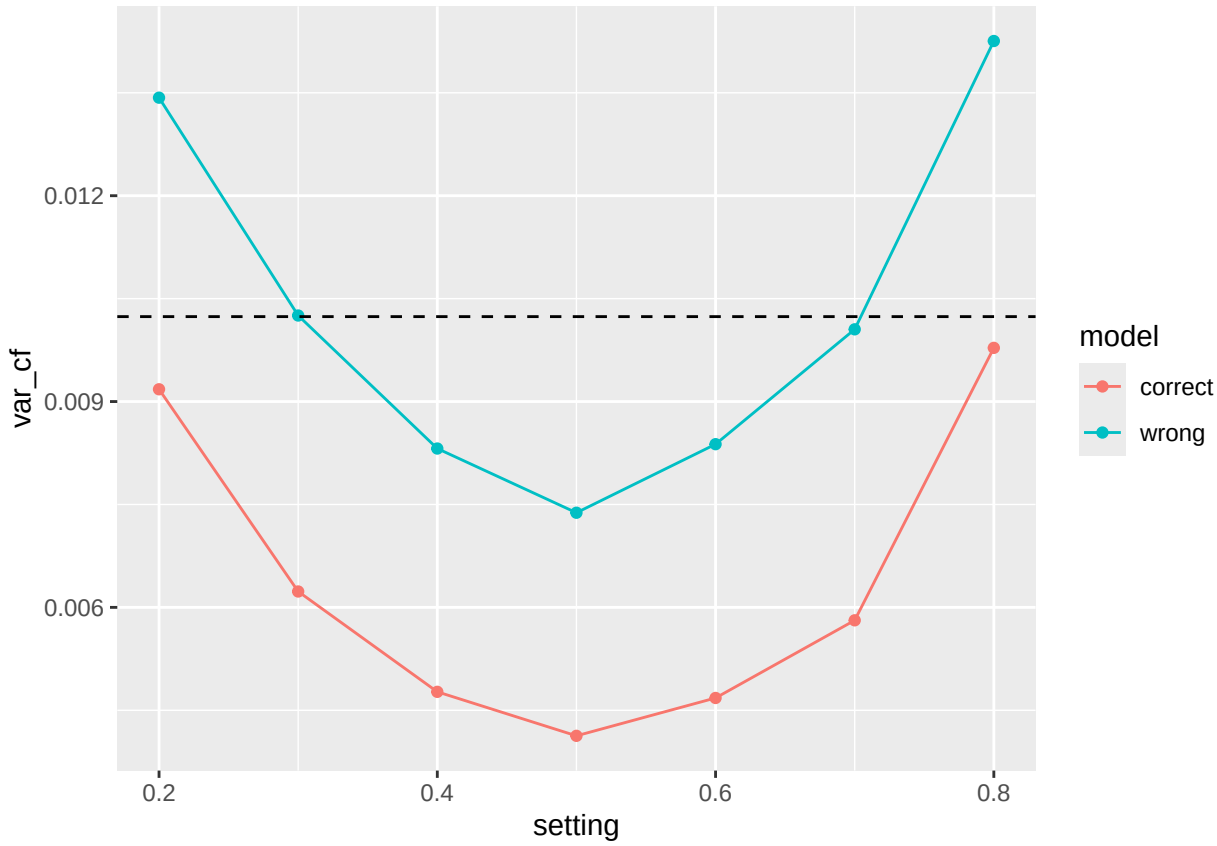
```r
combined_results_wrong_model = do.call(rbind, lapply(1:length(split_setting), function(i) {
  # Add a "setting" column to each data frame
  result_wrong_model[[i]]$setting = split_setting[i]
  return(result_wrong_model[[i]])
}))
```

plot variance comparisons

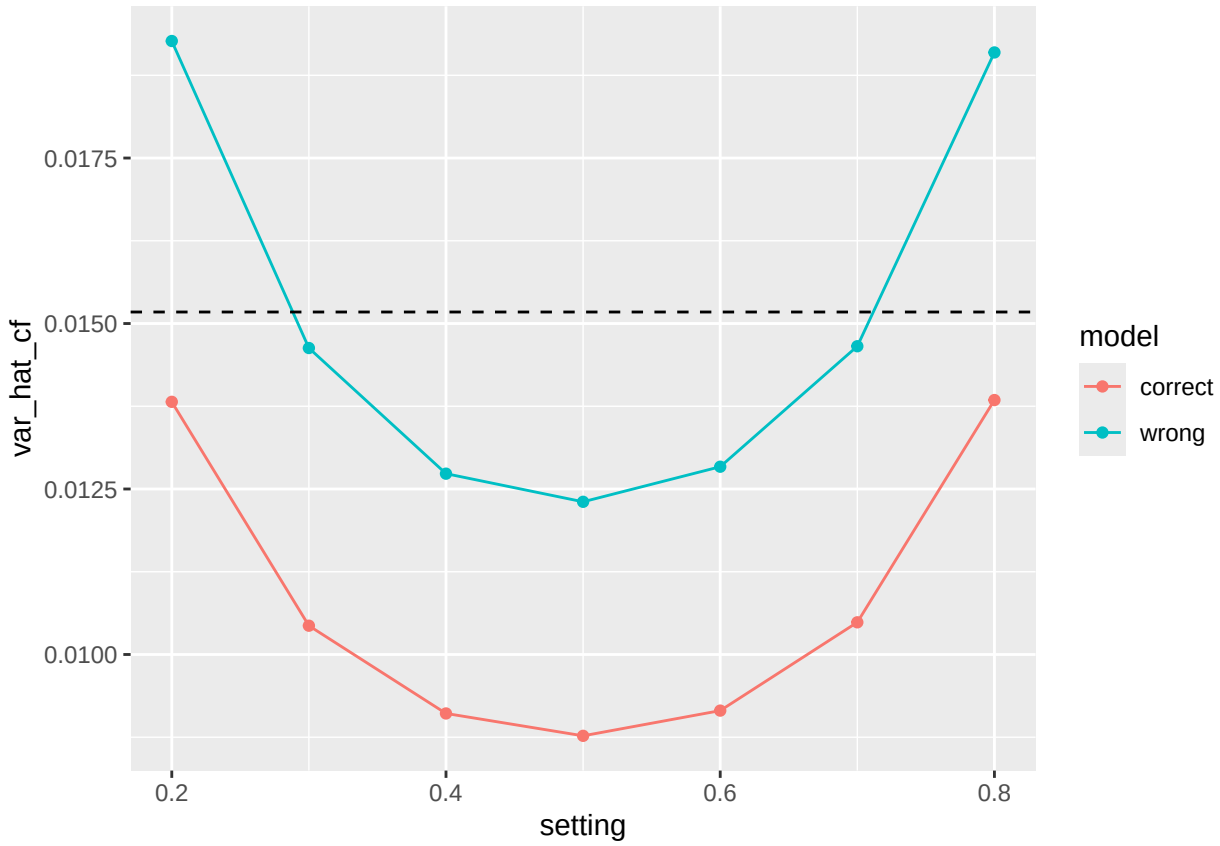
```r
# get mean variance without cross-fitting
var_0 = var(c(combined_results_correct_model$tau_hat_0,
              combined_results_wrong_model$tau_hat_0))
var_hat_0 = mean(c(combined_results_correct_model$var_hat_0,
                   combined_results_wrong_model$var_hat_0))

# get summary data for variance
summary_var = rbind(
  combined_results_correct_model %>%
    group_by(setting) %>%
    summarize(
      var_cf = var(tau_hat_cf),
      var_hat_cf = mean(var_hat_cf)
    ) %>%
    mutate(model = "correct"),
  combined_results_wrong_model %>%
    group_by(setting) %>%
    summarize(
      var_cf = var(tau_hat_cf),
      var_hat_cf = mean(var_hat_cf)
    ) %>%
    mutate(model = "wrong")
)
# %>%
#   pivot_longer(
#     cols = c("var_cf", "var_hat_cf"),
#     names_to = "var_type",
#     values_to = "value"
#   )


summary_var %>% ggplot() +
  geom_line(aes(x = setting, y = var_cf, col = model)) +
  geom_point(aes(x = setting, y = var_cf, col = model)) +
  geom_hline(yintercept = var_0, lty = 2)
```

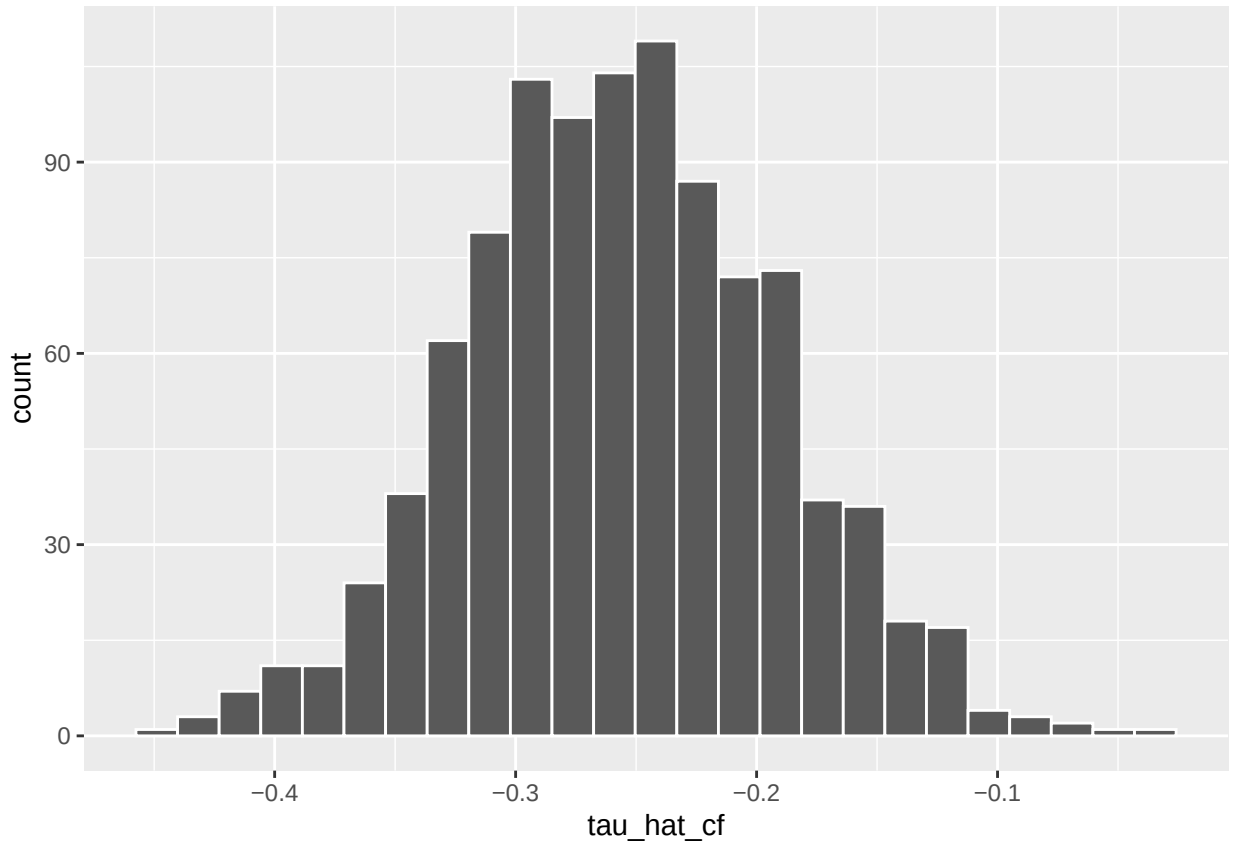



```
summary_var %>% ggplot() +
  geom_line(aes(x = setting, y = var_hat_cf, col = model)) +
  geom_point(aes(x = setting, y = var_hat_cf, col = model)) +
  geom_hline(yintercept = var_hat_0, lty = 2)
```

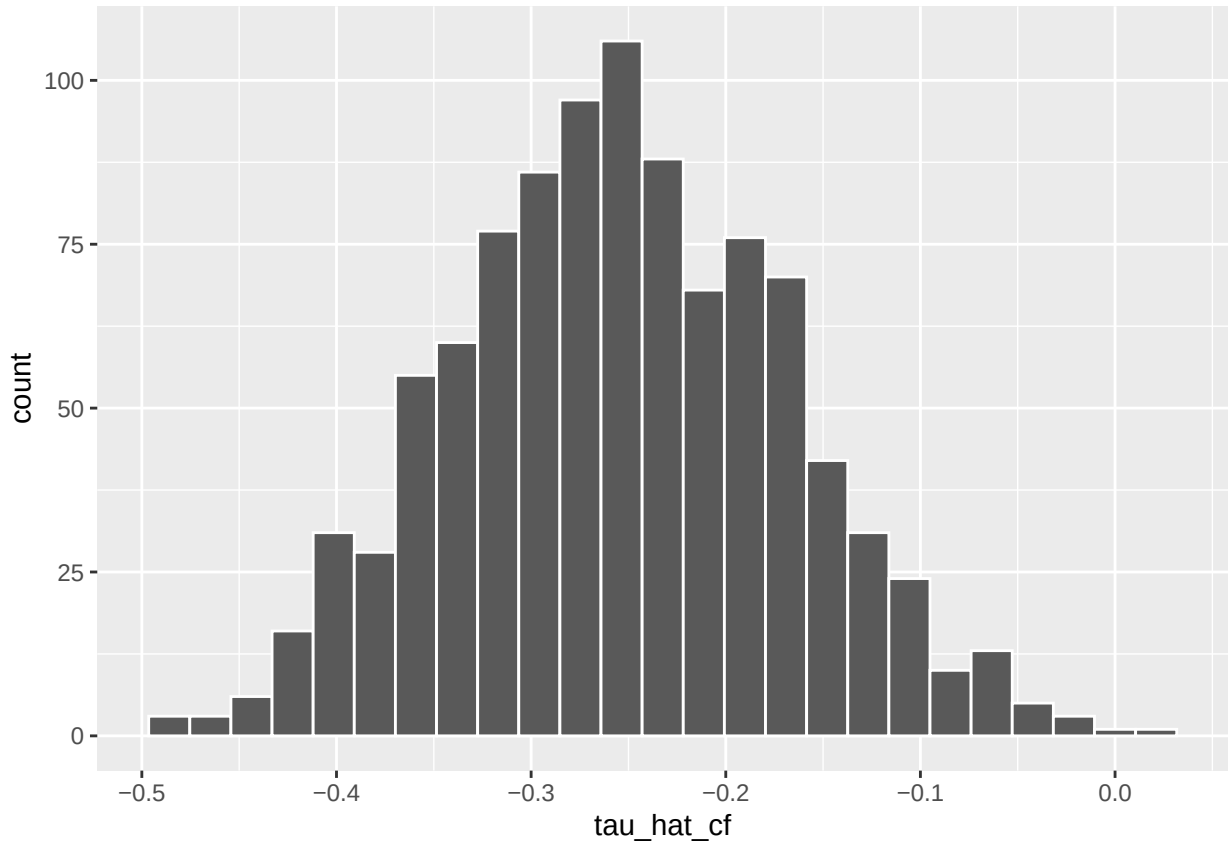


Distribution check

```
# At the optimal variance case, the distribution is asymptotically normal
result_correct_model[[4]] %>%
  ggplot() +
  geom_histogram(aes(x = tau_hat_cf), col = "white", bins = 25)
```

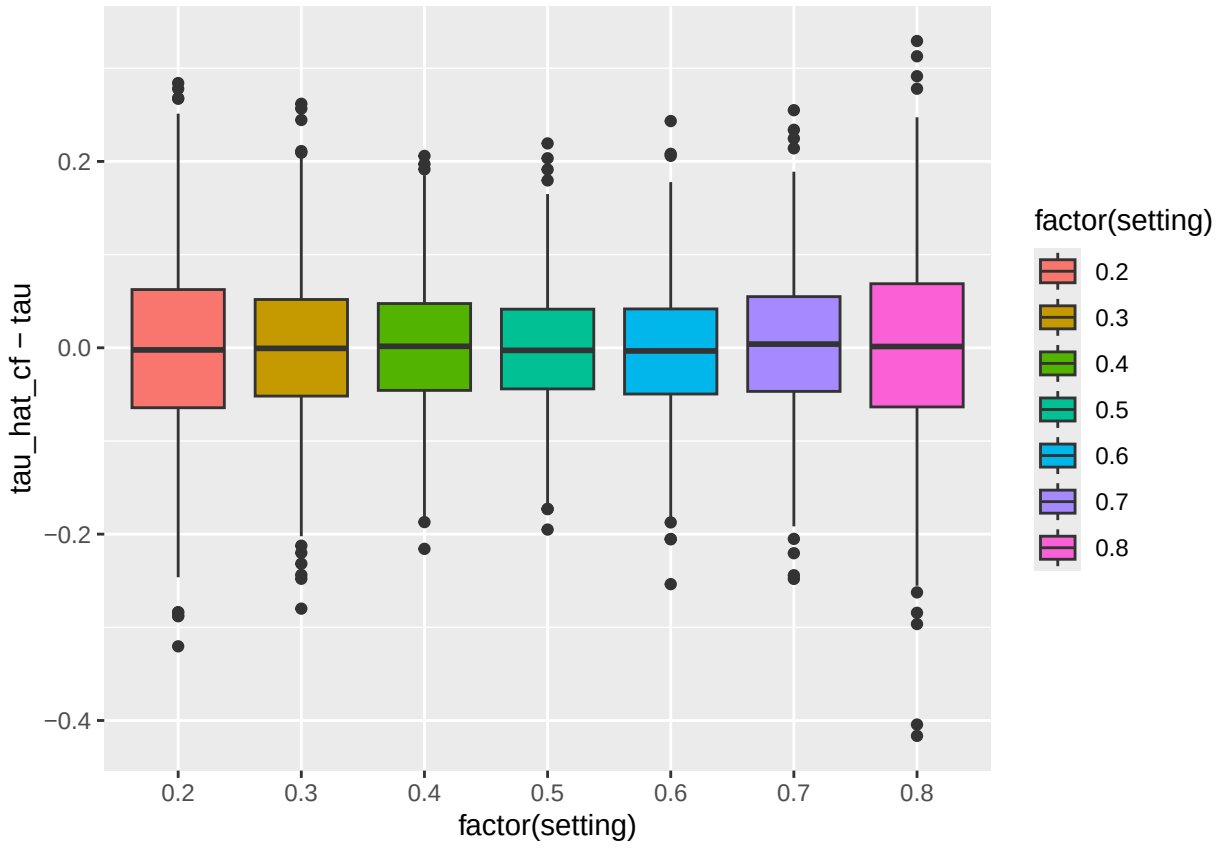



```
result_wrong_model[[4]] %>%
  ggplot() +
  geom_histogram(aes(x = tau_hat_cf), col = "white", bins = 25)
```



bias check

```
combined_results_correct_model %>%
  ggplot() +
  geom_boxplot(aes(x=factor(setting), y=tau_hat_cf-tau, fill = factor(setting)))
```



Setting 2:

Setting 3: Comparison with the strategy of FZ under Bernoulli trial

Setting 4: SRE setting and Matched Pair setting