CounterfactualFairness Replication

Aylin Seibold

2024-02-25

Load required packages

```
packages <- c("dplyr", "caret", "rstan", "ggplot2", "patchwork", "cowplot",
              "grid")
if(length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}
invisible(lapply(packages, library, character.only = TRUE))
```

```
##
## Attache Paket: 'dplyr'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union

## Lade nötiges Paket: ggplot2

## Lade nötiges Paket: lattice

## Lade nötiges Paket: StanHeaders

##
## rstan version 2.32.5 (Stan version 2.32.2)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)

## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
##
## Attache Paket: 'cowplot'

## Das folgende Objekt ist maskiert 'package:patchwork':
##
##     align_plots
```

Import and modify data

```
# Read the raw data
raw_data <- read.csv("law_data.csv")

# Select relevant columns
law <- dplyr::select(raw_data, race, sex, LSAT, UGPA, region_first, ZFYA,
                     sander_index, first_pf)

# Exclude certain region
law <- law[law$region_first != "PO",]
law$region_first <- factor(law$region_first)

# Convert categorical variables to numeric and give protected attributes
# their own column
law$amerind <- as.numeric(law$race == "Amerindian")
law$asian   <- as.numeric(law$race == "Asian")
law$black   <- as.numeric(law$race == "Black")
law$hisp    <- as.numeric(law$race == "Hispanic")
law$mexican <- as.numeric(law$race == "Mexican")
law$other   <- as.numeric(law$race == "Other")
law$puerto  <- as.numeric(law$race == "Puertorican")
law$white   <- as.numeric(law$race == "White")

law$female    <- as.numeric(law$sex == 1)
law$male      <- as.numeric(law$sex == 2)

# Define protected attributes
sense_cols <- c("amerind", "asian", "black", "hisp", "mexican", "other",
                "puerto", "white", "male", "female")
```

Data Partitioning

```
# Split data into training and testing sets using package ´caret´
set.seed(0)
trainIndex <- createDataPartition(law$first_pf, p = .8,
                                  list = FALSE,
                                  times = 1)
lawTrain <- law[trainIndex,]
lawTest  <- law[-trainIndex,]

n <- nrow(lawTrain)
ne <- nrow(lawTest)
```

```r
# Round LSAT scores
lawTrain$LSAT <- round(lawTrain$LSAT)
lawTest$LSAT <- round(lawTest$LSAT)
```

Stan Models

```r
# Set up the training data for the Stan model
# Training latent variable U (Know)
law_stan_train <- list(N = n, # Number of samples in the training data
                       K = length(sense_cols), # Number of protected attributes
                       # Matrix of protected attributes
                       a = data.matrix(lawTrain[,sense_cols]),
                       ugpa = lawTrain[,c("UGPA")],  # UGPA scores
                       lsat = lawTrain[,c("LSAT")], # LSAT scores
                       zfya = lawTrain[,c("ZFYA")]) # ZFYA scores

# Check if the trained Stan model exists (It can be found here:
# https://syncandshare.lrz.de/getlink/fiCwJ5Zx7PfbMW32LYfB3Q/)

if(file.exists("law_school_l_stan_train.rds")) {

  # If the model exists, load it
  la_law_train <- readRDS("law_school_l_stan_train.rds")

} else {

  # If the model does not exist, fit a new Stan model
  fit_law_train <- stan(file = "law_school_train.stan", # Stan model file
                        data = law_stan_train,  # Training data
                        iter = 2000,  # Number of iterations
                        chains = 1, verbose = TRUE) # Number of chains

  # Extract the fitted model parameters
  la_law_train <- extract(fit_law_train, permuted = TRUE)

  # Save the trained model
  saveRDS(la_law_train, file = "law_school_l_stan_train.rds")
}

# Calculate the mean of U parameters from the trained model
U_TRAIN   <- colMeans(la_law_train$u)

# Calculate the means of certain parameters from the trained model
ugpa0      <- mean(la_law_train$ugpa0)
eta_u_ugpa <- mean(la_law_train$eta_u_ugpa)
eta_a_ugpa <- colMeans(la_law_train$eta_a_ugpa)

lsat0      <- mean(la_law_train$lsat0)
eta_u_lsat <- mean(la_law_train$eta_u_lsat)
eta_a_lsat <- colMeans(la_law_train$eta_a_lsat)
```

```r
SIGMA_G <- mean(la_law_train$sigma_g)


# Same procedure with test data using the trained parameters of UGPA and LSAT
# Set up the test data for the Stan model using the trained parameters from the
# training data
law_stan_test <- list(N = ne, K = length(sense_cols),
                      a = data.matrix(lawTest[,sense_cols]),
                      ugpa = lawTest[,c("UGPA")],
                      lsat = lawTest[,c("LSAT")],
                      # Mean of ugpa0 from training data
                      ugpa0 = ugpa0,
                      # Mean of eta_u_ugpa from training data
                      eta_u_ugpa = eta_u_ugpa,
                      # Mean of eta_a_ugpa from training data
                      eta_a_ugpa = eta_a_ugpa,
                      # Mean of lsat0 from training data
                      lsat0 = lsat0,
                      # Mean of eta_u_lsat from training data
                      eta_u_lsat = eta_u_lsat,
                      # Mean of eta_a_lsat from training data
                      eta_a_lsat = eta_a_lsat,
                      # Mean of sigma_g from training data
                      sigma_g = SIGMA_G)

# Check if the test model exists
if(file.exists("law_school_l_stan_test.rds")) {
  la_law_test <- readRDS("law_school_l_stan_test.rds")
} else {
  fit_law_test <- stan(file = "law_school_only_u.stan",
                       data = law_stan_test, iter = 2000,
                       chains = 1, verbose = TRUE)
  la_law_test <- extract(fit_law_test, permuted = TRUE)
  saveRDS(la_law_test, file = "law_school_l_stan_test.rds")

}

# Calculate the mean of U parameters from the test model
U_TEST   <- colMeans(la_law_test$u)
```

Classifiers on data

Full Model

```r
# Convert the training data to data frames and add additional columns

# Convert protected attributes to a data frame
X_U <- as.data.frame(data.matrix(lawTrain[,sense_cols]))
X_U$ZFYA <- lawTrain$ZFYA  # Add ZFYA column to the data frame
X_U$LSAT <- lawTrain$LSAT   # Add LSAT column to the data frame
X_U$UGPA <- lawTrain$UGPA  # Add UGPA column to the data frame
```

```r
# Convert the test data to data frames and add additional columns
X_U_TE <- as.data.frame(data.matrix(lawTest[,sense_cols]))
X_U_TE$ZFYA <- lawTest$ZFYA
X_U_TE$LSAT <- lawTest$LSAT
X_U_TE$UGPA <- lawTest$UGPA

# Fit a logistic regression model on the training data using all variables
model_u <- lm(ZFYA ~ LSAT + UGPA + amerind + asian + black + hisp + mexican +
                other + puerto + white + male + female + 1, data=X_U)

# Make predictions on the test data using the fitted model
pred_u_te <- predict(model_u, newdata=X_U_TE)

# Calculate the RMSE for the test data
rmse_u_te <- sqrt( sum( (pred_u_te - X_U_TE$ZFYA)^2 ) / nrow(X_U_TE) )

# Print the RMSE for the unfair full model
print('unfair full model:')
```

```
## [1] "unfair full model:"
```

```r
print(rmse_u_te)
```

```
## [1] 0.8848115
```

Unaware Model

```r
# Fit a logistic regression model on the training data using only LSAT and UGPA
model_un <- lm(ZFYA ~ LSAT + UGPA + 1, data=X_U)

# Make predictions on the test data using the fitted model
pred_un_te <- predict(model_un, newdata=X_U_TE)

# Calculate the RMSE for the test data
rmse_un_te <- sqrt( sum( (pred_un_te - X_U_TE$ZFYA)^2 ) / nrow(X_U_TE) )

# Print the RMSE for the unfair full model
print('unfair unaware model:')
```

```
## [1] "unfair unaware model:"
```

```r
print(rmse_un_te)
```

```
## [1] 0.9064898
```

Fair K

```r
# Create data frames for the fair k model, including the predicted 'u'
# values and the observed 'ZFYA' values for training and testing data
X_F <- data.frame(u=U_TRAIN, ZFYA=lawTrain$ZFYA)
X_F_TE <- data.frame(u=U_TEST, ZFYA=lawTest$ZFYA)

# Fit a logistic regression model on the training data using U
model_f <- lm(ZFYA ~ u + 1, data=X_F)

# Make predictions on the test data using the fitted model
pred_f_te <- predict.glm(model_f, newdata=X_F_TE)

# Calculate the RMSE for the test data
rmse_f_te <- sqrt( sum( (pred_f_te - X_F_TE$ZFYA)^2 ) / nrow(X_F_TE) )

# Print the RMSE for the unfair full model
print('fair non-deterministic model:')
```

```
## [1] "fair non-deterministic model:"
```

```r
print(rmse_f_te)
```

```
## [1] 0.938474
```

Fair Add

```r
# Fit linear regression models to regress UGPA on race and sex, and LSAT on
# race and sex

# Train data
model_ugpa <- lm(UGPA ~ amerind + asian + black + hisp + mexican + other +
                   puerto + white + male + female + 1, data=lawTrain)
model_lsat <- lm(LSAT ~ amerind + asian + black + hisp + mexican + other +
                   puerto + white + male + female + 1, data=lawTrain)

# Calculate the residuals for UGPA/LSAT by subtracting the predicted UGPA/LSAT
# values from the observed UGPA/LSAT values
lawTrain$resid_UGPA = lawTrain$UGPA - predict(model_ugpa, newdata=lawTrain)
lawTrain$resid_LSAT = lawTrain$LSAT - predict(model_lsat, newdata=lawTrain)

# Fit a logistic regression model on the training data using residuals
model_det <- lm(ZFYA ~ resid_UGPA + resid_LSAT + 1, data=lawTrain)

# Fit linear regression models to regress UGPA on race and sex, and LSAT on race
# and sex

# Test data
model_ugpa_te <- lm(UGPA ~ amerind + asian + black + hisp + mexican + other +
                      puerto + white + male + female + 1, data=lawTest)
model_lsat_te <- lm(LSAT ~ amerind + asian + black + hisp + mexican + other +
                      puerto + white + male + female + 1, data=lawTest)
```

```r
# Calculate the residuals
lawTest$resid_UGPA = lawTest$UGPA - predict(model_ugpa_te, newdata=lawTest)
lawTest$resid_LSAT = lawTest$LSAT - predict(model_lsat_te, newdata=lawTest)

# Make predictions on the test data using the fitted model
pred_det_te <- predict(model_det, newdata=lawTest)

# Calculate the RMSE for the test data
rmse_det_te <- sqrt( sum( (pred_det_te - lawTest$ZFYA)^2 ) / nrow(lawTest) )

# Print the RMSE for the unfair full model
print('fair deterministic model:')
```

```
## [1] "fair deterministic model:"
```

```r
print(rmse_det_te)
```

```
## [1] 0.9311776
```

RMSE Table

```r
# Create a vector of RMSE values and a vector of corresponding model names
RMSE <- c(rmse_u_te, rmse_un_te, rmse_f_te, rmse_det_te)
Model <- c("Full", "Unaware", "Fair K", "Fair Add")

# Create a data frame to store the RMSE values and model names
RMSE.Table <- data.frame(Model,RMSE)
```

Boxplots

```r
data.te <- X_U_TE  # Set the test data

# Function to generate predictions and transform data for each model
data.fun <- function(model, data) {
  if (model == "Full") {              # Predictions for the full model
    data$predictions <- pred_u_te
  } else if (model == "Unaware") {    # Predictions for the unaware model
    data$predictions <- pred_un_te
  } else if (model == "FairK") {      # Predictions for the fair k model
    data$predictions <- pred_f_te
  } else if (model == "FairAdd") {    # Predictions for the fair Add model
    data$predictions <- pred_det_te
  }
  data <- data %>%

    # Transform the `black` column
    mutate(black = ifelse(black == 1, "black", black)) %>%
    # Transform the `asian` column
```

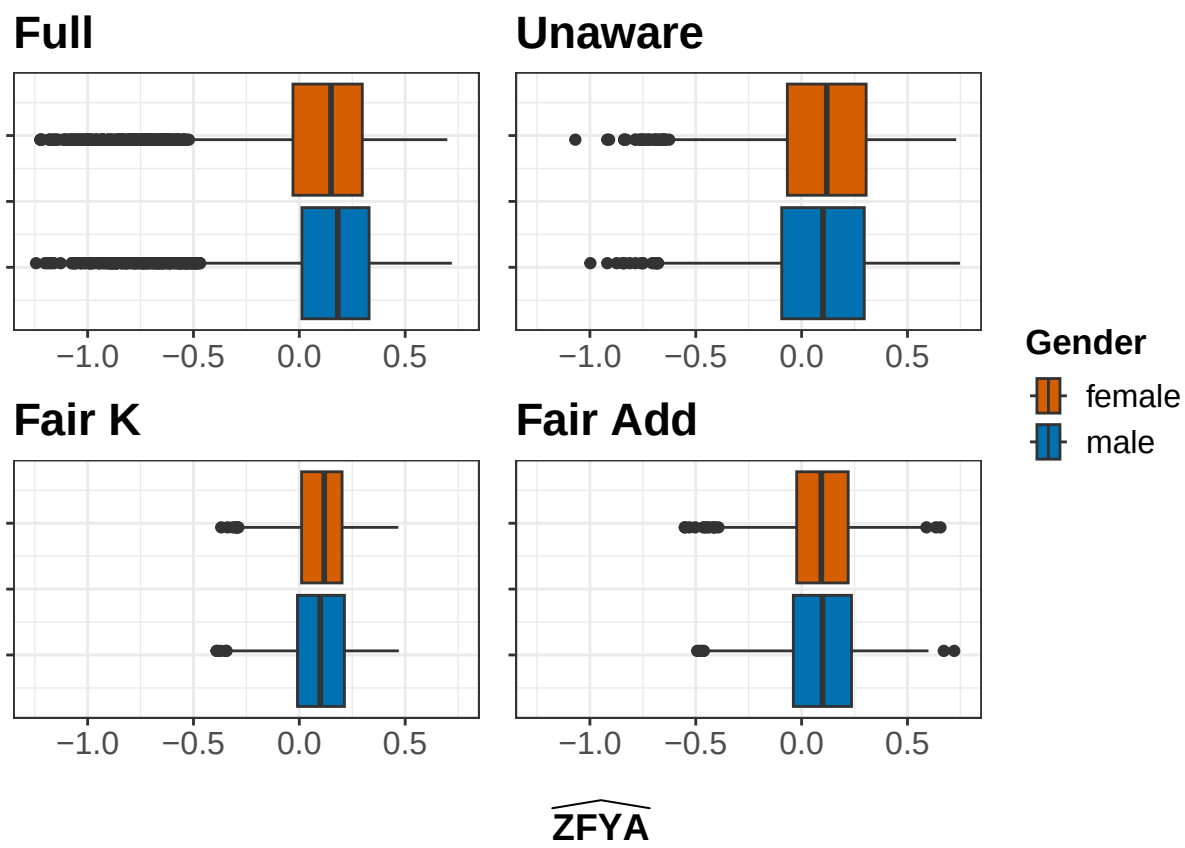
```
    mutate(asian = ifelse(asian == 1, "asian", asian)) %>%
    # Transform the `mexican` column
    mutate(mexican = ifelse(mexican == 1, "mexican", mexican))
  data <- data %>%
     # Combine columns
    mutate(combined_column = paste0(black, white, asian, mexican))
  return(data)
}

# Generate transformed data for each model
data.full <- data.fun("Full", data.te)
data.unaware <- data.fun("Unaware", data.te)
data.K <- data.fun("FairK", data.te)
data.Add <- data.fun("FairAdd", data.te)

# Function to create boxplots for gender
gender.plot.fun <- function(data) {
  ggplot(data, aes(x = predictions, fill = as.factor(female))) +
    geom_boxplot() +
    theme_bw() +
    guides(fill = guide_legend(title = "Gender")) +
    xlim(-1.25,0.75) +
    labs(title = NULL, ylab = NULL, xlab = NULL) +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.text = element_text(size = 12),
          legend.text = element_text(size = 12),
          legend.title = element_text(size = 13, face = "bold"),
          title = element_text(size = 14, face = "bold")) +
    scale_fill_manual(name = "Gender",
                      values = c( "1" = "#D55E00", "0" = "#0072B2"),
                      labels = c("1" = "female", "0" = "male"),
                      breaks = c("1", "0"))
}

# Function to create boxplots for race
race.plot.fun <- function(data) {
  ggplot(data, aes(x = predictions, fill = as.factor(combined_column))) +
    geom_boxplot() +
    theme_bw() +
    xlim(-1.25, 0.75) +
    guides(fill = guide_legend(title = "Race")) +
  labs(title = NULL, ylab = NULL, xlab = NULL) +
    theme(axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.text = element_text(size = 12),
          legend.text = element_text(size = 12),
          legend.title = element_text(size = 13, face = "bold"),
          title = element_text(size = 14, face = "bold")) +
    scale_fill_manual(name = "Race",
                      values = c("0100" = "#D55E00", "00asian0" = "#0072B2",
```

```
                                "black000" = "#009E73", "0000" = "#999999",
                                "000mexican" =  "#CC79A7"),
                     labels = c("0100" = "white", "00asian0" = "asian",
                                "black000" = "black", "0000" = "other",
                                "000mexican" =  "mexican"),
                     breaks = c("black000","0100", "00asian0", "000mexican",
                                "0000"))
}

# Create boxplots for gender for each model
plot1 <- gender.plot.fun(data.full) + labs(title = "Full")
plot2 <- gender.plot.fun(data.unaware) + labs(title = "Unaware")
plot3 <- gender.plot.fun(data.K) + labs(title = "Fair K")
plot4 <- gender.plot.fun(data.Add) + labs(title = "Fair Add")

# Create boxplots for race for each model
plot5 <- race.plot.fun(data.full) + labs(title = "Full")
plot6 <- race.plot.fun(data.unaware) + labs(title = "Unaware")
plot7 <- race.plot.fun(data.K) + labs(title = "Fair K")
plot8 <- race.plot.fun(data.Add) + labs(title = "Fair Add")

# Combine boxplots for gender for each model
combined.plot.gender <- plot1 + plot2 + plot3 + plot4 +
  plot_layout(ncol = 2, guides = "collect")

combined.plot.gender <- patchwork::patchworkGrob(combined.plot.gender)
```

```
## Warning: Removed 3 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

```
## Warning: Removed 9 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

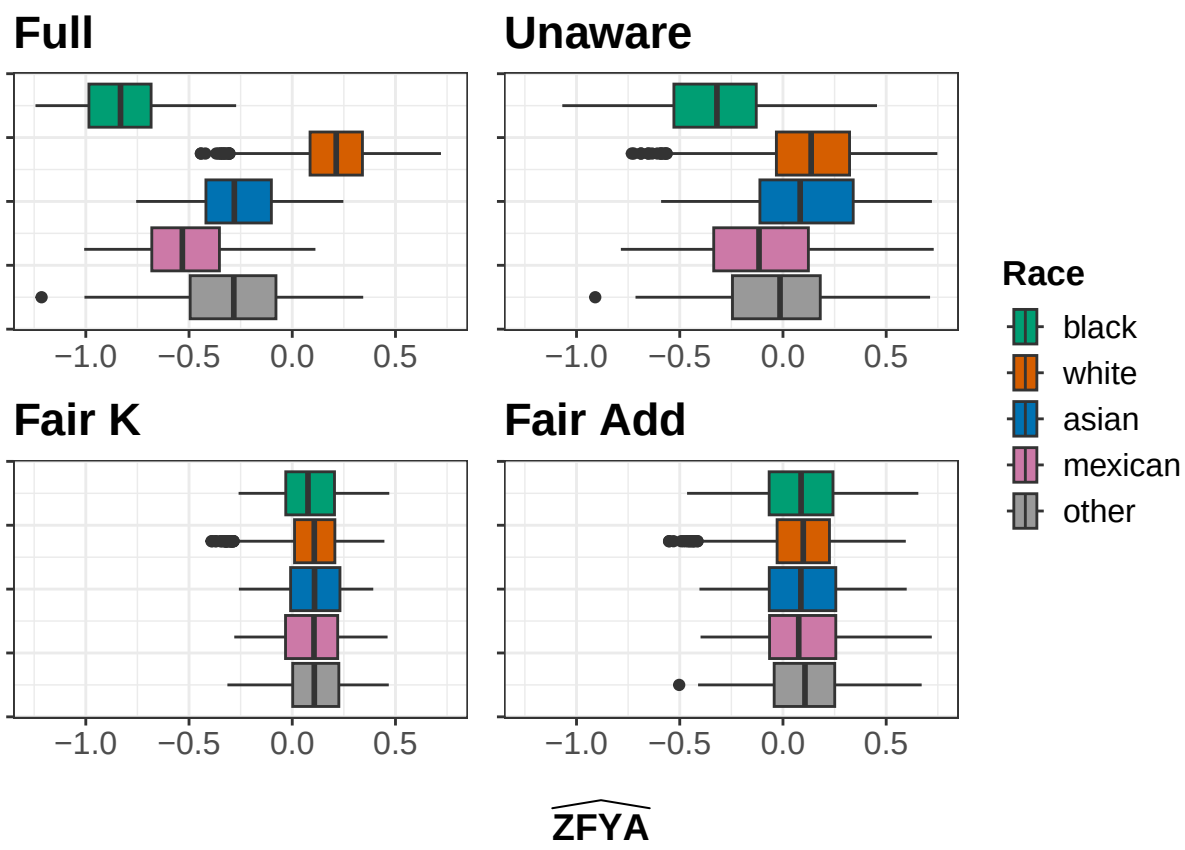
```
gridExtra::grid.arrange(combined.plot.gender,
                        bottom = textGrob(bquote(bold(widehat(ZFYA))),
                                          gp = gpar(fontsize = 14)))
```




```
# Combine boxplots for race for each model
combined.plot.race <- plot5 + plot6 + plot7 + plot8 +
  plot_layout(ncol = 2, guides = "collect")

combined.plot.race <- patchwork::patchworkGrob(combined.plot.race)
```

```
## Warning: Removed 3 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
## Removed 9 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```

```
gridExtra::grid.arrange(combined.plot.race,
                        bottom = textGrob(bquote(bold(widehat(ZFYA))),
                                          gp = gpar(fontsize = 14)))
```


Full
Unaware
Fair K
Fair Add
−1.0
−0.5
0.0
0.5
ZFYA
Race
black
white
asian
mexican
other