Assessing Balance In HTE ~ ICLR DB

```
rm(list=ls())
```

```
library(ggplot2)
library(tidyr)
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(dplyr)
```

```
##
## 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(grf)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x psych::%+%()    masks ggplot2::%+%()
## x psych::alpha()  masks ggplot2::alpha()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
require(gridExtra)
```

```
## Loading required package: gridExtra
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(xtable)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(sandwich)
```

```
# Auxiliary function to computes adjusted p-values
# following the Romano-Wolf method.
# For a reference, see http://ftp.iza.org/dp12845.pdf page 8
#  t.orig: vector of t-statistics from original model
#  t.boot: matrix of t-statistics from bootstrapped models
romano_wolf_correction <- function(t.orig, t.boot) {
  abs.t.orig <- abs(t.orig)
  abs.t.boot <- abs(t.boot)
  abs.t.sorted <- sort(abs.t.orig, decreasing = TRUE)

  max.order <- order(abs.t.orig, decreasing = TRUE)
  rev.order <- order(max.order)

  M <- nrow(t.boot)
  S <- ncol(t.boot)

  p.adj <- rep(0, S)
  p.adj[1] <- mean(apply(abs.t.boot, 1, max) > abs.t.sorted[1])
  for (s in seq(2, S)) {
    cur.index <- max.order[s:S]
    p.init <- mean(apply(abs.t.boot[, cur.index, drop=FALSE], 1, max) > abs.t.sorted[s])
    p.adj[s] <- max(p.init, p.adj[s-1])
  }
```

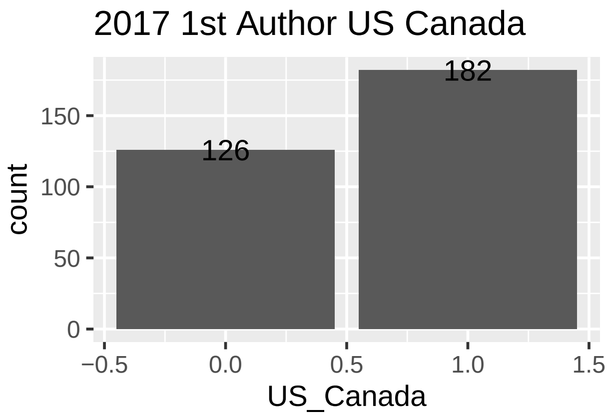
```
  p.adj[rev.order]
}

# Computes adjusted p-values for linear regression (lm) models.
#    model: object of lm class (i.e., a linear reg model)
#    indices: vector of integers for the coefficients that will be tested
#    cov.type: type of standard error (to be passed to sandwich::vcovHC)
#    num.boot: number of null bootstrap samples. Increase to stabilize across runs.
# Note: results are probabilitistic and may change slightly at every run.
#
# Adapted from the p_adjust from from the hdm package, written by Philipp Bach.
# https://github.com/PhilippBach/hdm_prev/blob/master/R/p_adjust.R
summary_rw_lm <- function(model, indices=NULL, cov.type="HC2", num.boot=10000) {

  if (is.null(indices)) {
    indices <- 1:nrow(coef(summary(model)))
  }
  # Grab the original t values.
  summary <- coef(summary(model))[indices,,drop=FALSE]
  t.orig <- summary[, "t value"]

  # Null resampling.
  # This is a trick to speed up bootstrapping linear models.
  # Here, we don't really need to re-fit linear regressions, which would be a bit slow.
  # We know that betahat ~ N(beta, Sigma), and we have an estimate Sigmahat.
  # So we can approximate "null t-values" by
  #  - Draw beta.boot ~ N(0, Sigma-hat) --- note the 0 here, this is what makes it a *null* t-value.
  #  - Compute t.boot = beta.boot / sqrt(diag(Sigma.hat))
  Sigma.hat <- vcovHC(model, type=cov.type)[indices, indices]
  se.orig <- sqrt(diag(Sigma.hat))
  num.coef <- length(se.orig)
  beta.boot <- mvrnorm(n=num.boot, mu=rep(0, num.coef), Sigma=Sigma.hat)
  t.boot <- sweep(beta.boot, 2, se.orig, "/")
  p.adj <- romano_wolf_correction(t.orig, t.boot)

  result <- cbind(summary[,c(1,2,4),drop=F], p.adj)
  colnames(result) <- c('Estimate', 'Std. Error', 'Orig. p-value', 'Adj. p-value')
  result
}
```

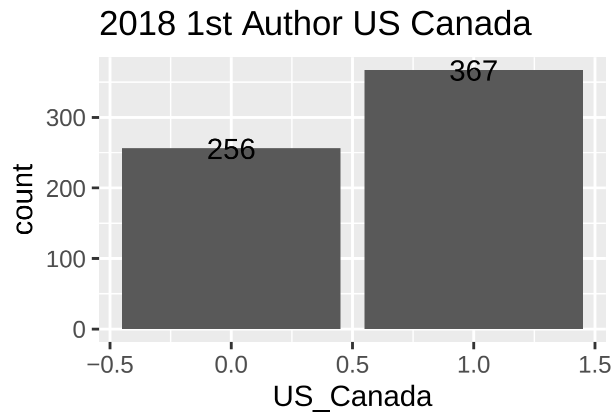
```
setwd('~/Mirror/github/aICLR/')

df_authors = read.csv("./data/database/outputs/df_authors.csv")
df_prestige = read.csv("./data/database/outputs/df_prestige.csv")
```

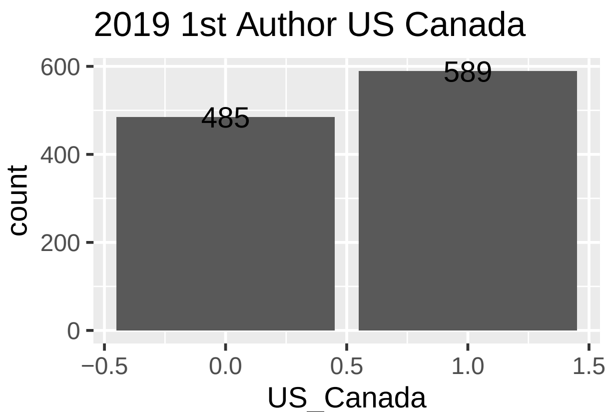
```
Authors_2017 = df_authors %>% group_by(author_id) %>%
  filter(conf_year == 2017, author_no == 1, current_position_flag == 1)

Authors_2018 = df_authors %>% group_by(author_id) %>%
  filter(conf_year == 2018, author_no == 1, current_position_flag == 1)

Authors_2019 = df_authors %>% group_by(author_id) %>%
  filter(conf_year == 2019, author_no == 1, current_position_flag == 1)
```

```r
g1 = ggplot(Authors_2017, aes(US_Canada)) +
  geom_bar() +
  ggtitle("2017 1st Author US Canada") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g2 = ggplot(Authors_2018, aes(US_Canada)) +
  geom_bar() +
  ggtitle("2018 1st Author US Canada") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g3 = ggplot(Authors_2019, aes(US_Canada)) +
  geom_bar() +
  ggtitle("2019 1st Author US Canada") +
      stat_count(geom = "text",
             aes(label = stat(count)),
             position="stack", colour="black")
grid.arrange(g1, g2, g3,ncol=2)
```

```
## Warning: `stat(count)` was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.

## Warning: Removed 118 rows containing non-finite values (`stat_count()`).
## Removed 118 rows containing non-finite values (`stat_count()`).

## Warning: Removed 221 rows containing non-finite values (`stat_count()`).
## Removed 221 rows containing non-finite values (`stat_count()`).

## Warning: Removed 228 rows containing non-finite values (`stat_count()`).
## Removed 228 rows containing non-finite values (`stat_count()`).
```

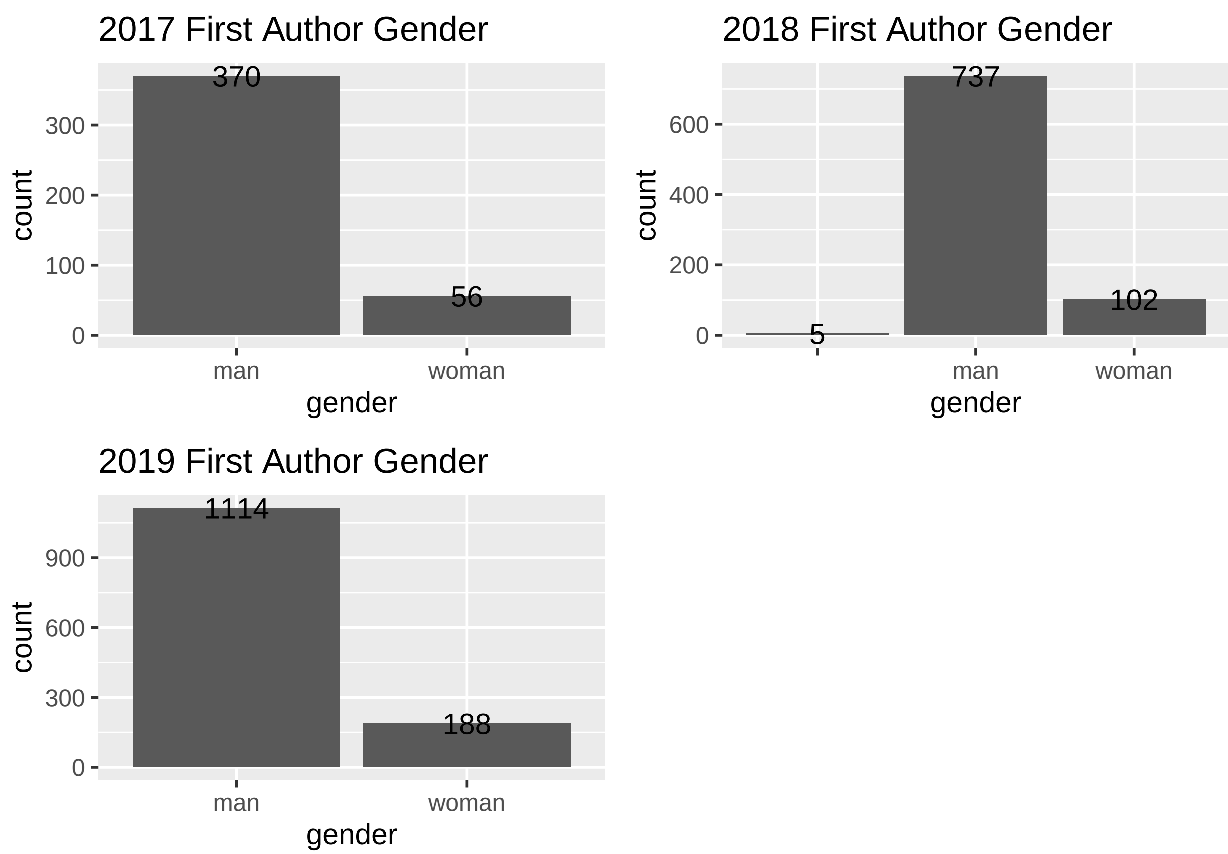






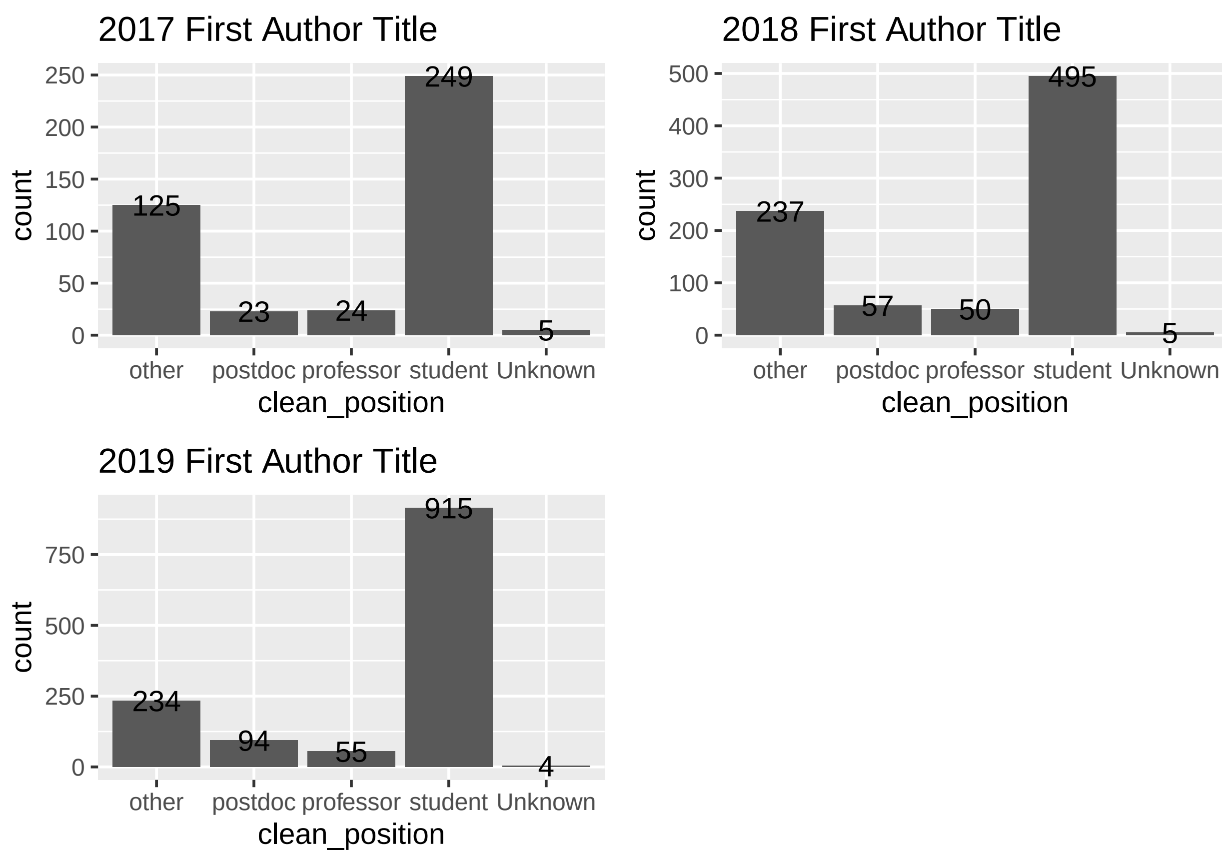

```
Authors_2017 = df_authors %>% filter(conf_year == 2017, author_no == 1, current_position_flag == 1)

Authors_2018 = df_authors %>% filter(conf_year == 2018, author_no == 1, current_position_flag == 1)

Authors_2019 = df_authors %>% filter(conf_year == 2019, author_no == 1, current_position_flag == 1)


g1 = ggplot(Authors_2017, aes(gender)) +
  geom_bar() +
  ggtitle("2017 First Author Gender") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g2 = ggplot(Authors_2018, aes(gender)) +
  geom_bar() +
  ggtitle("2018 First Author Gender") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g3 = ggplot(Authors_2019, aes(gender)) +
  geom_bar() +
  ggtitle("2019 First Author Gender") +
      stat_count(geom = "text",
             aes(label = stat(count)),
```

```
            position="stack", colour="black")

grid.arrange(g1, g2, g3,ncol=2)
```



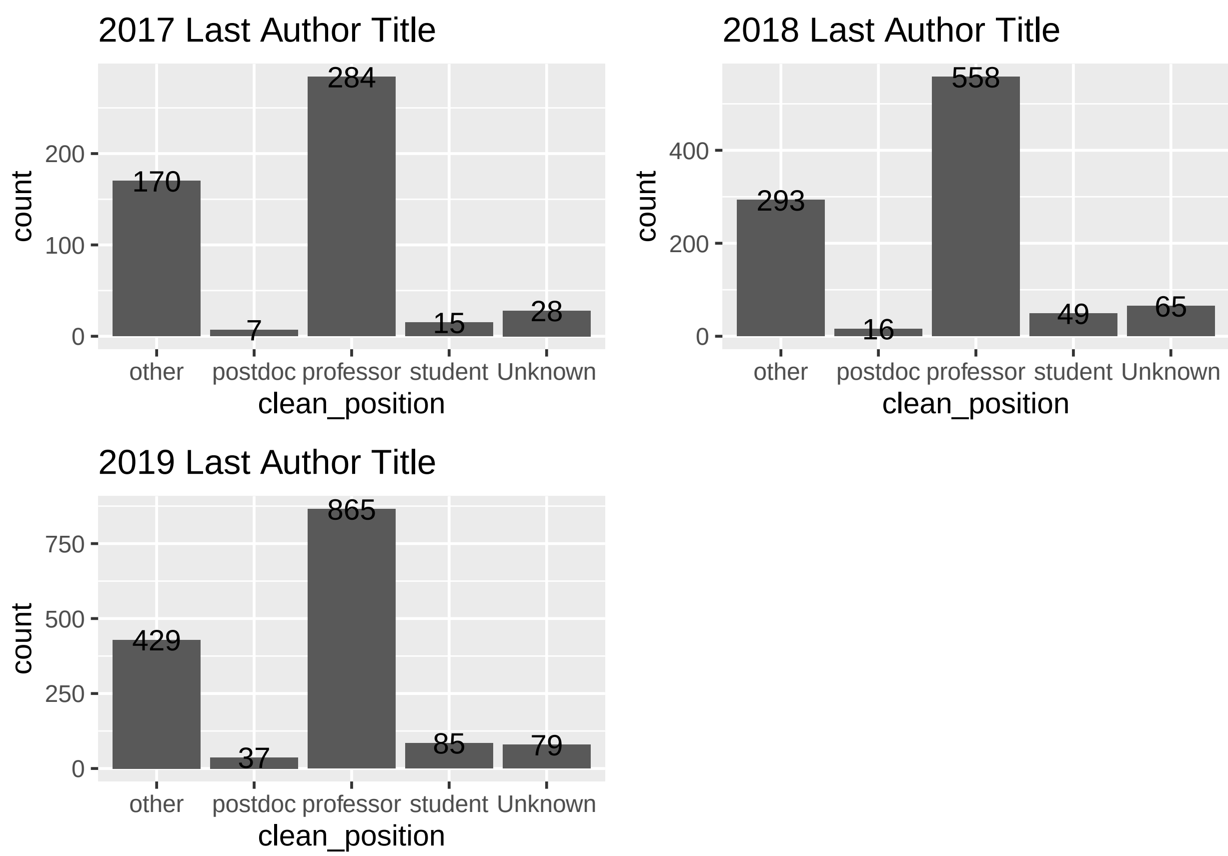

```
Authors_2017 = df_authors %>% filter(conf_year == 2017, author_no == 1, current_position_flag == 1)

Authors_2018 = df_authors %>% filter(conf_year == 2018, author_no == 1, current_position_flag == 1)

Authors_2019 = df_authors %>% filter(conf_year == 2019, author_no == 1, current_position_flag == 1)


g1 = ggplot(Authors_2017, aes(clean_position)) +
  geom_bar() +
  ggtitle("2017 First Author Title") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g2 = ggplot(Authors_2018, aes(clean_position)) +
  geom_bar() +
  ggtitle("2018 First Author Title") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")
```

```
g3 = ggplot(Authors_2019, aes(clean_position)) +
  geom_bar() +
  ggtitle("2019 First Author Title") +
      stat_count(geom = "text",
             aes(label = stat(count)),
             position="stack", colour="black")
grid.arrange(g1, g2, g3,ncol=2)
```




```
Authors_2017 = df_authors %>% group_by(submission_id) %>%
  top_n(1, author_no) %>%
  filter(conf_year == 2017, current_position_flag == 1)

Authors_2018 = df_authors %>% group_by(submission_id) %>%
  top_n(1, author_no) %>%
  filter(conf_year == 2018, current_position_flag == 1)

Authors_2019 = df_authors %>%
  group_by(submission_id) %>%
  top_n(1, author_no) %>%
  filter(conf_year == 2019, current_position_flag == 1)


g1 = ggplot(Authors_2017, aes(clean_position)) +
  geom_bar() +
  ggtitle("2017 Last Author Title") +
```

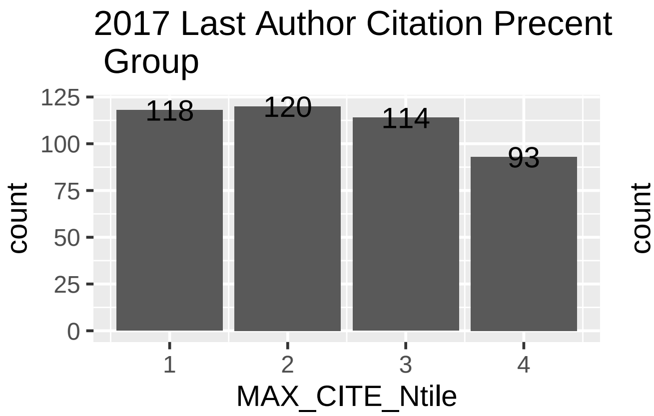
```
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g2 = ggplot(Authors_2018, aes(clean_position)) +
  geom_bar() +
  ggtitle("2018 Last Author Title") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g3 = ggplot(Authors_2019, aes(clean_position)) +
  geom_bar() +
  ggtitle("2019 Last Author Title") +
      stat_count(geom = "text",
             aes(label = stat(count)),
             position="stack", colour="black")
grid.arrange(g1, g2, g3,ncol=2)
```

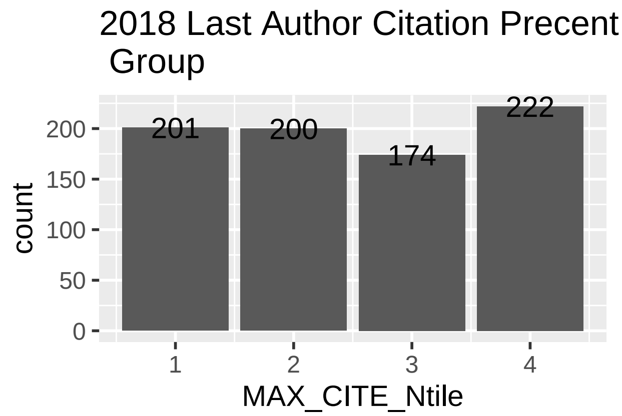


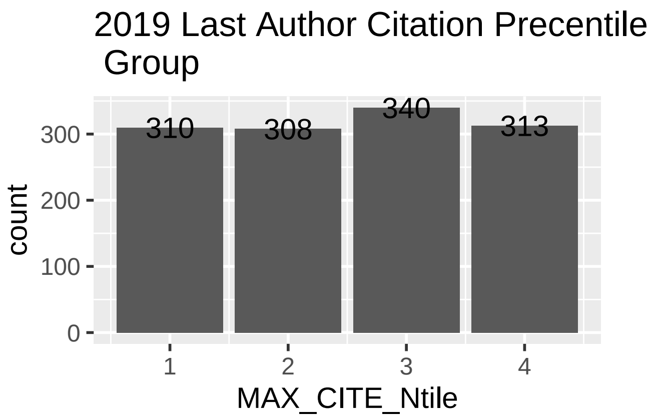

```
df_prestige = df_prestige %>% mutate(MAX_CITE_Ntile = ntile(MAX_CITE, 4))
df_prestige_2017 = df_prestige  %>%
  filter(conf_year ==2017)


df_prestige_2018 = df_prestige %>%
```

```r
  filter(conf_year ==2018)

df_prestige_2019 = df_prestige %>%
  filter(conf_year ==2019)

g1 = ggplot(df_prestige_2017, aes(MAX_CITE_Ntile)) +
  geom_bar() +
  ggtitle("2017 Last Author Citation Precentile \n Group") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g2 = ggplot(df_prestige_2018, aes(MAX_CITE_Ntile)) +
  geom_bar() +
  ggtitle("2018 Last Author Citation Precentile \n Group") +
  stat_count(geom = "text",
       aes(label = stat(count)),
       position="stack", colour="black")

g3 = ggplot(df_prestige_2019, aes(MAX_CITE_Ntile)) +
  geom_bar() +
  ggtitle("2019 Last Author Citation Precentile \n Group") +
      stat_count(geom = "text",
            aes(label = stat(count)),
            position="stack", colour="black")
grid.arrange(g1, g2, g3,ncol=2)
```

```
## Warning: Removed 45 rows containing non-finite values (`stat_count()`).
## Removed 45 rows containing non-finite values (`stat_count()`).

## Warning: Removed 114 rows containing non-finite values (`stat_count()`).
## Removed 114 rows containing non-finite values (`stat_count()`).

## Warning: Removed 148 rows containing non-finite values (`stat_count()`).
## Removed 148 rows containing non-finite values (`stat_count()`).
```







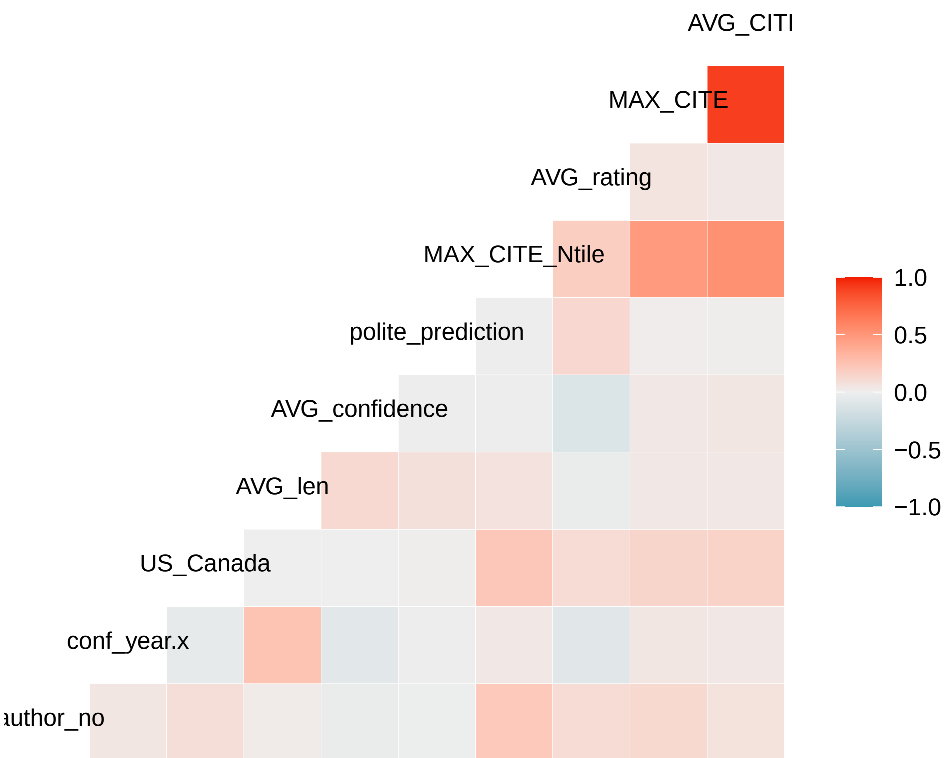

```
features = c("author_no","conf_year.x","US_Canada","clean_position","gender","AVG_len" , "AVG_confidenc
df_combined = merge(df_authors, df_prestige,by.x = 'submission_id', by.y ='id') %>% filter(current_posi

df_combined_limited_features = df_combined[features]

df_authors %>% filter(author_no ==1 & current_position_flag==1) %>% group_by(conf_year) %>% summarize(n
```

```
## # A tibble: 3 x 2
##   conf_year     n
##       <int> <int>
## 1      2017   426
## 2      2018   844
## 3      2019  1302
```

```
ggcorr(df_combined_limited_features,hjust = 0.5,vjust=.2, size = 3) + ggplot2::labs(title = "Pearson Co
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning in ggcorr(df_combined_limited_features, hjust = 0.5, vjust = 0.2, : data
## in column(s) 'clean_position', 'gender' are not numeric and were ignored
```

Pearson Correlation Matrix




```r
df_combined_limited_features_first_author = df_combined_limited_features %>% filter(author_no == 1 & co

# Fit the full model
full.model <- lm(polite_prediction ~., data = na.omit(df_combined_limited_features_first_author))
library(MASS)
step.model <- stepAIC(full.model, direction = "both",
                      trace = FALSE)
summary(step.model)
```

```
##
## Call:
## lm(formula = polite_prediction ~ conf_year.x + AVG_len + MAX_CITE_Ntile +
##     AVG_rating + MAX_CITE + AVG_CITE, data = na.omit(df_combined_limited_features_first_author))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43707 -0.21804  0.01914  0.11689  0.81997
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.288e+02  3.618e+01   3.561 0.000390 ***
## conf_year.x    -6.386e-02  1.793e-02  -3.561 0.000390 ***
## AVG_len         1.810e-04  4.638e-05   3.902 0.000103 ***
## MAX_CITE_Ntile -1.483e-02  9.104e-03  -1.629 0.103649
## AVG_rating      4.601e-02  6.911e-03   6.658    5e-11 ***
```

```
## MAX_CITE        3.373e-06  1.535e-06   2.197 0.028261 *
## AVG_CITE       -1.029e-05  4.930e-06  -2.086 0.037260 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2362 on 841 degrees of freedom
## Multiple R-squared:  0.08147,    Adjusted R-squared:  0.07492
## F-statistic: 12.43 on 6 and 841 DF,  p-value: 1.882e-13
```

```
full.model <- lm(polite_prediction ~., data = na.omit(df_combined_limited_features_first_author))
summary(full.model)
```

```
##
## Call:
## lm(formula = polite_prediction ~ ., data = na.omit(df_combined_limited_features_first_author))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42777 -0.21542  0.01786  0.11429  0.82106
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.285e+02  3.645e+01   3.525 0.000447 ***
## author_no                        NA         NA      NA       NA
## conf_year.x              -6.375e-02  1.806e-02  -3.530 0.000439 ***
## US_Canada                 1.637e-02  1.732e-02   0.945 0.344882
## clean_positionpostdoc     5.581e-02  4.654e-02   1.199 0.230821
## clean_positionprofessor   9.041e-02  4.569e-02   1.979 0.048190 *
## clean_positionstudent     4.268e-02  3.625e-02   1.177 0.239340
## clean_positionUnknown     3.723e-02  8.347e-02   0.446 0.655664
## genderman                 4.037e-02  1.069e-01   0.378 0.705725
## genderwoman               4.432e-02  1.089e-01   0.407 0.684049
## AVG_len                   1.754e-04  4.678e-05   3.750 0.000189 ***
## AVG_confidence            8.023e-03  1.621e-02   0.495 0.620797
## MAX_CITE_Ntile           -1.602e-02  9.301e-03  -1.722 0.085402 .
## AVG_rating                4.620e-02  7.133e-03   6.477  1.6e-10 ***
## MAX_CITE                  3.443e-06  1.540e-06   2.235 0.025666 *
## AVG_CITE                 -1.022e-05  4.952e-06  -2.063 0.039413 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2365 on 833 degrees of freedom
## Multiple R-squared:  0.08802,    Adjusted R-squared:  0.07269
## F-statistic: 5.743 on 14 and 833 DF,  p-value: 8.95e-11
```

```
full.model <- lm(AVG_rating ~. + conf_year.x*MAX_CITE_Ntile, data = na.omit(df_combined_limited_feature
summary(full.model)
```

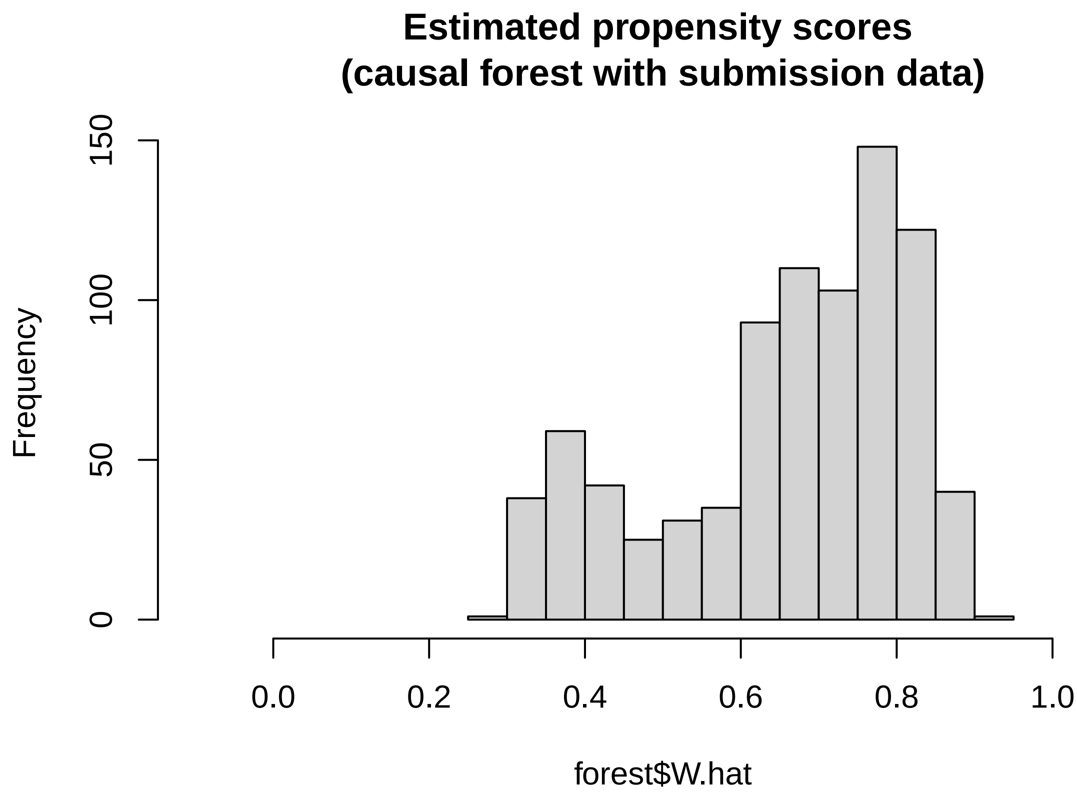
```
##
## Call:
## lm(formula = AVG_rating ~ . + conf_year.x * MAX_CITE_Ntile, data = na.omit(df_combined_limited_featu
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.9514 -0.8060 -0.0361  0.7773  3.4613
##
## Coefficients: (1 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -1.871e+02  4.015e+02  -0.466   0.6414
## author_no                         NA         NA      NA       NA
## conf_year.x                9.585e-02  1.990e-01   0.482   0.6302
## US_Canada                  1.401e-01  8.205e-02   1.708   0.0881 .
## clean_positionpostdoc     -7.634e-02  2.211e-01  -0.345   0.7300
## clean_positionprofessor   -6.766e-02  2.172e-01  -0.311   0.7555
## clean_positionstudent     -1.325e-02  1.720e-01  -0.077   0.9386
## clean_positionUnknown      5.793e-01  3.954e-01   1.465   0.1433
## genderman                  2.166e-01  5.068e-01   0.427   0.6693
## genderwoman                1.008e-02  5.164e-01   0.020   0.9844
## AVG_len                   -1.711e-04  2.236e-04  -0.765   0.4443
## AVG_confidence            -4.694e-01  7.532e-02  -6.232 7.29e-10 ***
## polite_prediction          1.032e+00  1.605e-01   6.431 2.14e-10 ***
## MAX_CITE_Ntile             1.145e+02  1.491e+02   0.768   0.4429
## MAX_CITE                   4.842e-06  7.341e-06   0.660   0.5097
## AVG_CITE                  -4.493e-05  2.349e-05  -1.913   0.0561 .
## conf_year.x:MAX_CITE_Ntile -5.661e-02  7.391e-02  -0.766   0.4439
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.121 on 832 degrees of freedom
## Multiple R-squared:  0.1463, Adjusted R-squared:  0.131
## F-statistic: 9.509 on 15 and 832 DF,  p-value: < 2.2e-16
```

```r
df_submissions_1718 = df_combined_limited_features_first_author %>% filter(conf_year.x %in% c(2017,2018
df_submissions_1718 = df_submissions_1718 %>% mutate(W = if_else(conf_year.x==2017,0,1))

covariates = c("US_Canada","clean_position","gender","AVG_len" ,"AVG_confidence","MAX_CITE","polite_pre
#covariates =  c("AVG_len" ,"MAX_CITE_Ntile","polite_prediction","AVG_confidence")


df_submissions_1718 = na.omit(df_submissions_1718)
XX <- model.matrix(formula(paste0("~", paste0(covariates, collapse="+"))), data=df_submissions_1718)

set.seed(1)

forest <- causal_forest(
              X=XX,
              W=df_submissions_1718[,"W"],
              Y=df_submissions_1718[,"AVG_rating"]
              #,num.trees = 100
              )


forest.ate <- average_treatment_effect(forest)


forest.ate
```

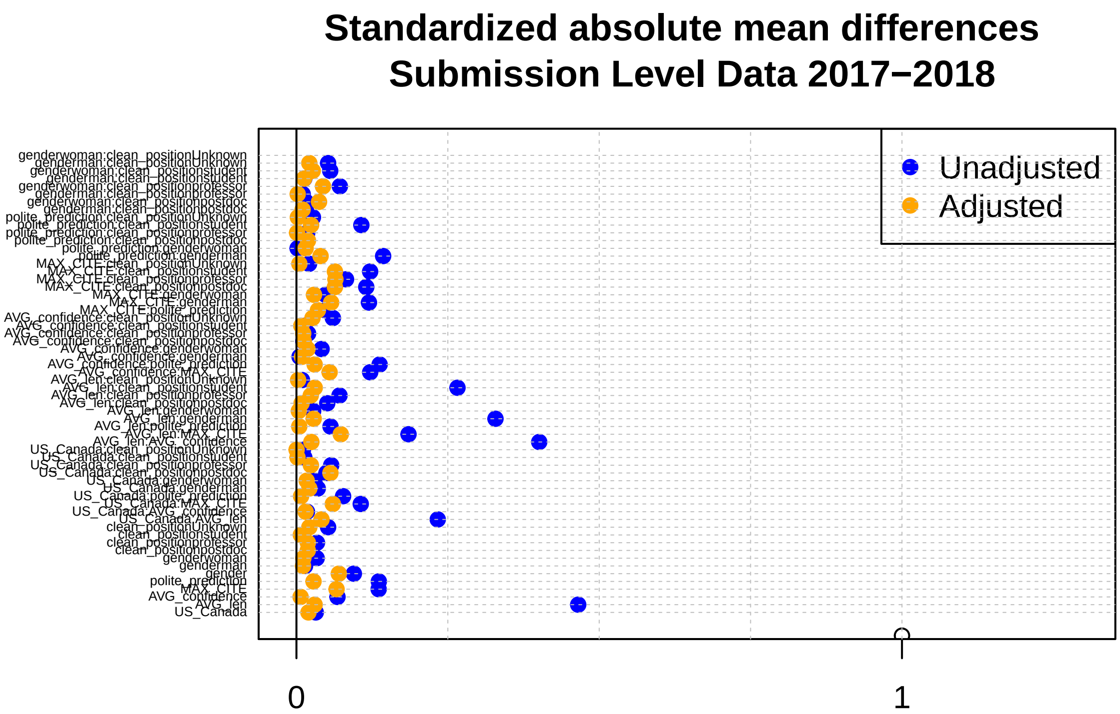
```
##    estimate     std.err
## -0.06920760  0.09428644
```

```
hist(forest$W.hat, main="Estimated propensity scores \n(causal forest with submission data)", xlim=c(-.
```




```
covariates = c("US_Canada","AVG_len" ,"AVG_confidence","MAX_CITE","polite_prediction","gender","clean_p
# Here, adding covariates and their interactions, though there are many other possibilities.
fmla <- formula(paste("~ 0 +", paste(apply(expand.grid(covariates, covariates), 1, function(x) paste0(x

# Using the propensity score estimated above
#check to see if you are using ATE causal forest ICLR_analysis_Submission_1718
e.hat <- forest$W.hat

XX <- model.matrix(fmla, df_submissions_1718)
W <- df_submissions_1718[,"W"]
pp <- ncol(XX)

# Unadjusted covariate means, variances and standardized abs mean differences
means.treat <- apply(XX[W == 1,], 2, mean)
means.ctrl <- apply(XX[W == 0,], 2, mean)
abs.mean.diff <- abs(means.treat - means.ctrl)

var.treat <- apply(XX[W == 1,], 2, var)
var.ctrl <- apply(XX[W == 0,], 2, var)
std <- sqrt(var.treat + var.ctrl)
```

```
# Adjusted covariate means, variances and standardized abs mean differences
means.treat.adj <- apply(XX*W/e.hat, 2, mean)
means.ctrl.adj <- apply(XX*(1-W)/(1-e.hat), 2, mean)
abs.mean.diff.adj <- abs(means.treat.adj - means.ctrl.adj)

var.treat.adj <- apply(XX*W/e.hat, 2, var)
var.ctrl.adj <- apply(XX*(1-W)/(1-e.hat), 2, var)
std.adj <- sqrt(var.treat.adj + var.ctrl.adj)

# Plotting
#png(file = "balance_ICLR_analysis_Submission_1718_ATE.png")
par(oma=c(0,4,0,0))
plot(-2, xaxt="n", yaxt="n", xlab="", ylab="", xlim=c(-.01, 1.3), ylim=c(0, pp+1), main="Standardized al
axis(side=1, at=c(-1, 0, 1), las=1)
lines(abs.mean.diff / std, seq(1, pp), type="p", col="blue", pch=19)
lines(abs.mean.diff.adj / std.adj, seq(1, pp), type="p", col="orange", pch=19)
legend("topright", c("Unadjusted", "Adjusted"), col=c("blue", "orange"), pch=19)
abline(v = seq(0, 1, by=.25), lty = 2, col = "grey", lwd=.5)
abline(h = 1:pp,  lty = 2, col = "grey", lwd=.5)
mtext(colnames(XX), side=2, cex=0.42, at=1:pp, padj=.4, adj=1, col="black", las=1, line=.3)
abline(v = 0)
```

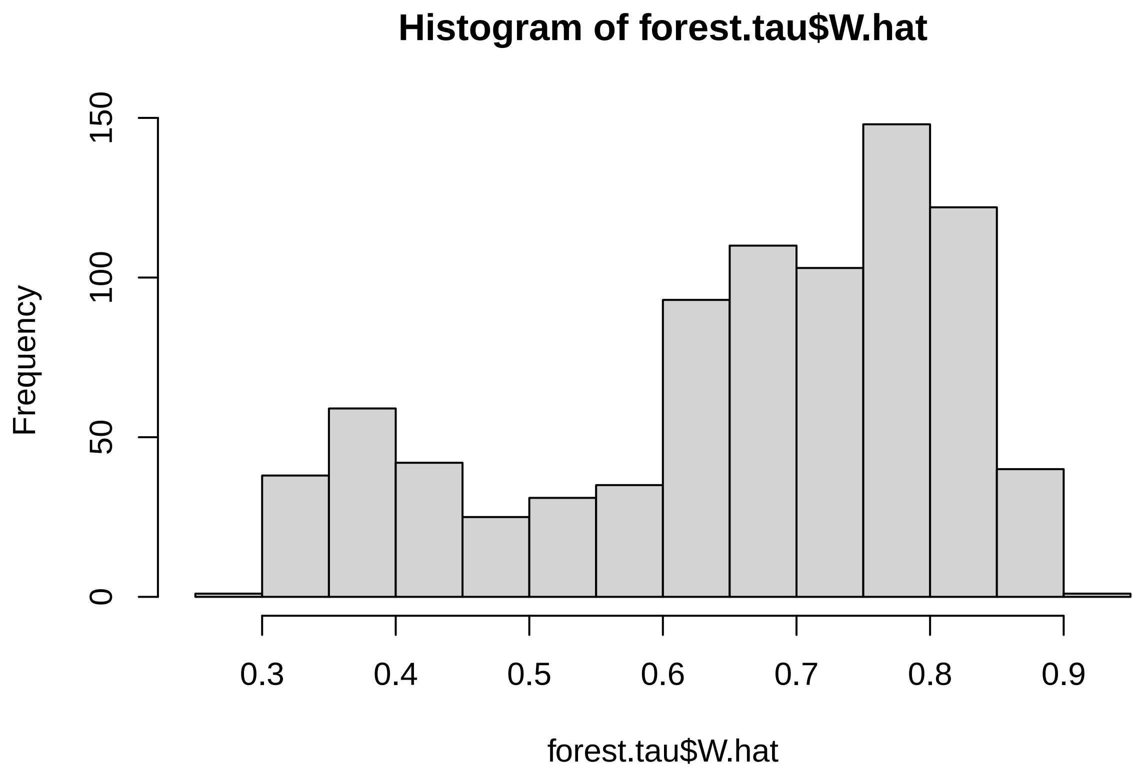



```
set.seed(1)
group = "US_Canada"
```

```r
covariates = c("US_Canada","clean_position","gender","AVG_len" ,"AVG_confidence","MAX_CITE","polite_pre

XX <- model.matrix(formula(paste0("~", paste0(covariates, collapse="+"))), data=df_submissions_1718)
W=df_submissions_1718[,"W"]
Y=df_submissions_1718[,"AVG_rating"]



forest.tau <- causal_forest(XX, Y, W)

tau.hat <- predict(forest.tau)$predictions
m.hat <- forest.tau$Y.hat  # E[Y|X] estimates
e.hat <- forest.tau$W.hat  # e(X) := E[W|X] estimates (or known quantity)
tau.hat <- forest.tau$predictions  # tau(X) estimates

# Predicting mu.hat(X[i], 1) and mu.hat(X[i], 0) for obs in held-out sample
# Note: to understand this, read equations 6-8 in this vignette
# https://grf-labs.github.io/grf/articles/muhats.html
mu.hat.0 <- m.hat - e.hat * tau.hat        # E[Y|X,W=0] = E[Y|X] - e(X)*tau(X)
mu.hat.1 <- m.hat + (1 - e.hat) * tau.hat  # E[Y|X,W=1] = E[Y|X] + (1 - e(X))*tau(X)

# Compute AIPW scores
aipw.scores <- tau.hat + W / e.hat * (Y -  mu.hat.1) - (1 - W) / (1 - e.hat) * (Y -  mu.hat.0)

# Estimate average treatment effect conditional on group membership
fmla <- formula(paste0('aipw.scores ~ factor(', group, ')'))
ols <- lm(fmla, data=transform(df_submissions_1718[covariates], aipw.scores=aipw.scores))

summary_rw_lm(ols)
```

```
##                       Estimate Std. Error Orig. p-value Adj. p-value
## (Intercept)          0.0271899  0.1493715     0.8556038       0.8574
## factor(US_Canada)1 -0.1602845  0.1926107     0.4055490       0.5562
```

```r
hist(forest.tau$W.hat)
```




```
test_calibration(forest.tau)
```

```
##
## Best linear fit using forest predictions (on held-out data)
## as well as the mean forest prediction as regressors, along
## with one-sided heteroskedasticity-robust (HC3) SEs:
##
##                                Estimate Std. Error t value Pr(>t)
## mean.forest.prediction           1.0499     1.2544  0.8370 0.2014
## differential.forest.prediction  -0.9320     0.8974 -1.0385 0.8503
```