Inference

Andri, Eden, Veronica

11/19/2023

```
library(dplyr)
source('dataProcessing.R')
library(estimatr)
library(randomForest)
library(latex2exp)
library(ggplot2)
dlexp = loadData()
```

```
# Difference in means
meanDiff <- function(z, y){
  ZY = data.frame(z)
  ZY$y = y
  treat = filter(ZY, z == 1)
  ctrl = filter(ZY, z == 0)

  return (mean(treat$y) - mean(ctrl$y))
}

# FRT
frtSim <- function(z, y, M, testStat, tObserved){
  pVals = vector(, M) # for p-value convergence
  pValCnt = 0
  for (i in 1:M){
    set.seed(i)
    zSim = sample(z)
    t = testStat(zSim, y)
    if (t >= tObserved){
      pValCnt = pValCnt + 1
    }
    pVals[i] = pValCnt/i
  }

  return (pVals)
}
```

```
# Neymanian inference
s_2_z <- function(data, z){
  sub_ = filter(data, Z == z)
  n_z = nrow(sub_)

  return (c(var(sub_$Y), n_z))
}
```

```r
# conservative variance estimate for Neyman
vHat = function(data){
  s_2_1 = s_2_z(data, 1)
  s_2_0 = s_2_z(data, 0)

  v_hat = (s_2_1[1]/s_2_1[2]) + (s_2_0[1]/s_2_0[2])
  return (v_hat)
}

# normal approx 95% confidence interval
ci <- function(that, vhat){
  return (c(that - 1.96*sqrt(vhat), that + 1.96*sqrt(vhat)))
}
```

```r
# discard those who did not get party cue
dParty = filter(dlexp, party.cues == 1, !is.na(pid3))

# run difference in means
tObs = meanDiff(dParty$Z, dParty$Y)
print(tObs)
```

```
## [1] 0.2696496
```

```r
# run frt
pVals = frtSim(dParty$Z, dParty$Y, 10000, meanDiff, tObs)
pVal = pVals[length(pVals)] # neg index does not work?
print(pVal)
```

```
## [1] 0.0041
```

```r
# get neymanian variance estimate and confidence interval
v_hat = vHat(dParty)
print(v_hat)
```

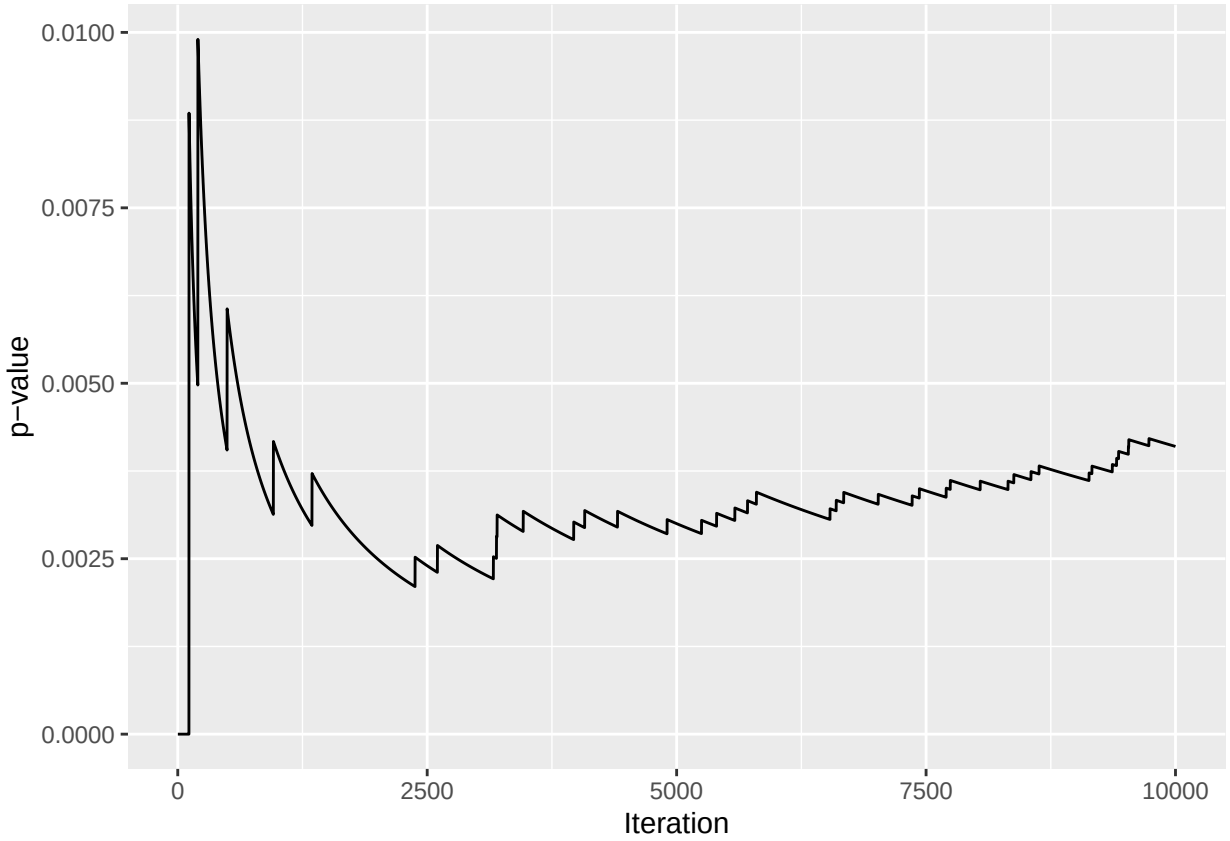
```
## [1] 0.01021915
```

```r
print(ci(tObs, v_hat))
```

```
## [1] 0.0715136 0.4677856
```

```r
# plot p-value convergence for FRT
dfP <- data.frame(Index = seq_along(pVals), pVals = pVals)

ggplot(dfP, aes(x = Index, y = pVals)) +
  geom_line() +
  labs(x = "Iteration", y = "p-value")
```




```
# run neymanian inference for each party/group separately

runParty = function(data, pid){
  dat = filter(data, pid3 == pid)
  tau_hat = meanDiff(dat$Z, dat$Y)
  v_hat = vHat(dat)
  ci_ = ci(tau_hat, v_hat)
  return (c(tau_hat, ci_))
}

# plot confidence intervals for each party/group on same plot
confIntPlot = function(dem, ind, rep, est_expr){
  df = rbind(dem, ind, rep)
  df <- as.data.frame(df)

  new_column_names <- c("tau_hat", "ci_lower", "ci_upper")
  colnames(df) <- new_column_names
  df$politicalParty = c("Democrat", "Independent", "Republican")

  ggplot(df, aes(politicalParty, tau_hat)) + geom_point() +
    geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper)) +
    xlab("Political Party") +
    ylab(TeX(est_expr)) +
    coord_flip()
}
```

```
(neymDem = runParty(dParty, 1)) # dem
```

```
## [1]  0.18523355 -0.09183471  0.46230181
```

```
(neymInd = runParty(dParty, 2)) # ind
```

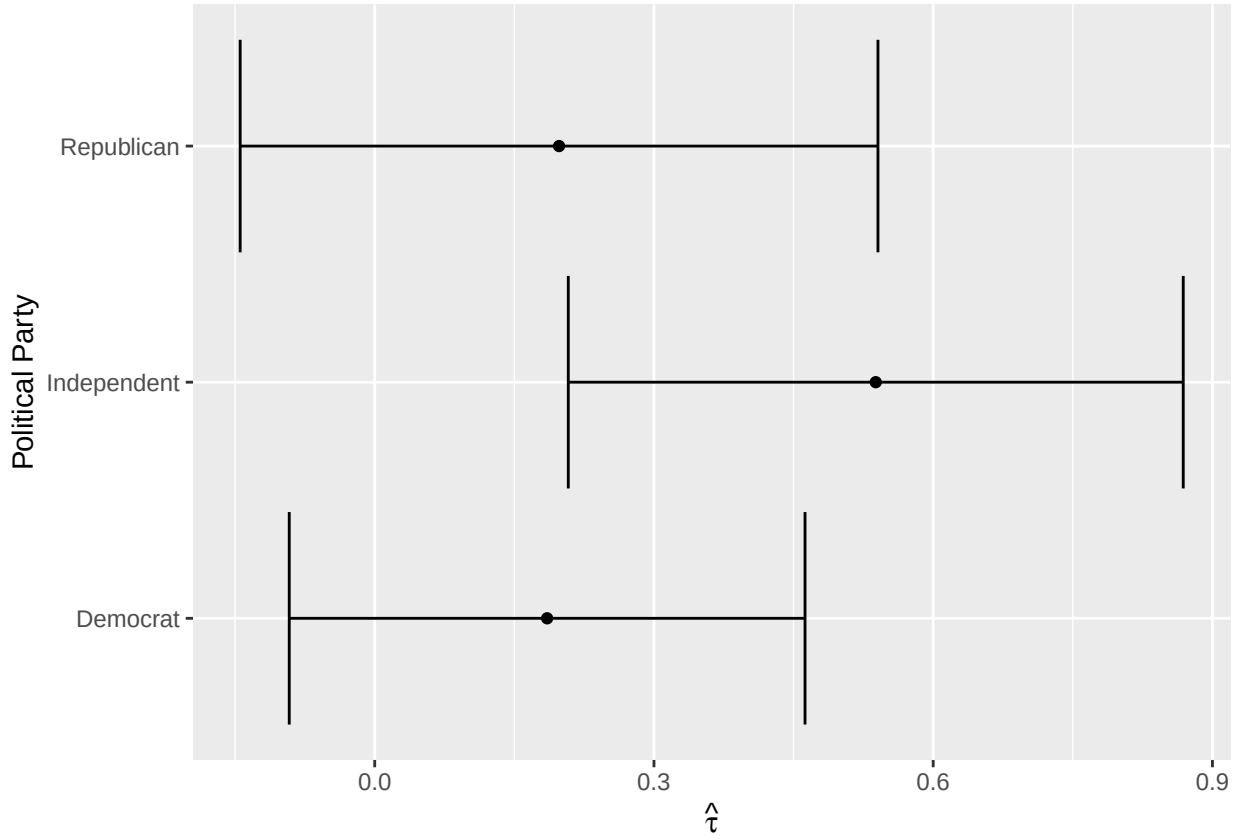
```
## [1] 0.5382901 0.2079570 0.8686231
```

```
(neymRep = runParty(dParty, 3)) # rep
```

```
## [1]  0.1980549 -0.1445401  0.5406500
```

```
# No surprises here, same results as paper


confIntPlot(neymDem, neymInd, neymRep, "$\\hat{\\tau}$")
```




```
dParty$Y_factor = as.factor(dParty$Y)

# shift by centering of other data
centerVars = function(dataCenter, dataShift){
  vars = c("income", "educ4", "age", "white") # covariates included in model, to shift
  data_ = dataShift
  for (var in vars){
    data_[, var] <- data_[, var] - mean(dataCenter[, var])
  }
  return (data_)
}
```

```r
# Generalized Lin's Estimator with random forest model

# Using classification to predict and then interpret as continuous for variance te/var calc
trainModel = function(data){
  data = droplevels(data)
  model = randomForest(Y_factor ~ income + educ4 + age + white, data = data)
  return (model)
}

# predict for x
# training data: data that model was trained on
predict_calibrate_cross = function(model,  x, leftout_data, training_data){
  # shift left-out data based on centering for training data
  x_shifted = centerVars(training_data, x)
  leftout_data_shifted = centerVars(training_data, leftout_data)

  n_d = nrow(leftout_data)
  mu_hat = as.numeric(predict(model, x_shifted))
  l_pred = as.numeric(predict(model, leftout_data_shifted))

  l_pred_shift = sum((leftout_data$Y) - l_pred)/n_d

  mu_out = mu_hat + l_pred_shift
  return (mu_out)
}


# t_data: training data
# l_data: left-out data
tau_pred_cross = function(control_model, treatment_model,
                          control_t_data, treatment_t_data,
                          control_l_data, treatment_l_data){

  n_t = nrow(control_t_data) + nrow(treatment_t_data)

  tau = (sum(treatment_t_data$Y) +
           sum(predict_calibrate_cross(treatment_model, control_t_data, treatment_l_data, treatment_t_da
           sum(control_t_data$Y) -
           sum(predict_calibrate_cross(control_model, treatment_t_data, control_l_data, control_t_data)

  return (tau)
}

t_pred_cross_total = function(control_model_1, control_model_2,
                              treatment_model_1, treatment_model_2,
                              control_data_1, control_data_2,
                              treatment_data_1, treatment_data_2){

  n_1 = nrow(control_data_1) + nrow(treatment_data_1)
  n_2 = nrow(control_data_2) + nrow(treatment_data_2)
  n = n_1 + n_2

  t_out = (tau_pred_cross(control_model_2, treatment_model_2,
```

```
                          control_data_2, treatment_data_2,
                          control_data_1, treatment_data_1)*(n_1/n)) +
    (tau_pred_cross(control_model_1, treatment_model_1,
                    control_data_1, treatment_data_1,
                    control_data_2, treatment_data_2)*(n_2/n))

  return (t_out)
}


get_tau_cross = function(data){
  set.seed(0102)
  split_ind = sample(seq_len(nrow(data)), size = floor(0.5*nrow(data)))

  d1 = data[split_ind, ]
  d1_centered = centerVars(d1, d1)

  d2 = data[-split_ind, ]
  d2_centered = centerVars(d2, d2)

  d1_ctrl = filter(d1, Z == 0)
  d1_ctrl_centered = filter(d1_centered, Z == 0)
  d1_treatment = filter(d1, Z == 1)
  d1_treatment_centered = filter(d1_centered, Z == 1)

  d2_ctrl = filter(d2, Z == 0)
  d2_ctrl_centered = filter(d2_centered, Z == 0)
  d2_treatment = filter(d2, Z == 1)
  d2_treatment_centered = filter(d2_centered, Z == 1)

  ctrl_model_1 = trainModel(d1_ctrl_centered)
  treatment_model_1 = trainModel(d1_treatment_centered)

  ctrl_model_2 = trainModel(d2_ctrl_centered)
  treatment_model_2 = trainModel(d2_treatment_centered)


  t_pred = t_pred_cross_total(ctrl_model_1, ctrl_model_2,
                              treatment_model_1, treatment_model_2,
                              d1_ctrl, d2_ctrl,
                              d1_treatment, d2_treatment)

  return (t_pred)

}

tau_hat_all = get_tau_cross(dParty)
print(tau_hat_all) # treatment effect estimate
```

```
## [1] 0.1876604
```

```
# bootstrap variance, this takes a long time to run for generalized Lin's
boostrapTau = function(data, tau_function, M){
  t = vector(, M)
```

```
  for (i in 1:M){
    set.seed(i)
    data_sample = sample_n(data, nrow(data), replace = TRUE)
     t[i] = tau_function(data_sample)
  }
  return (t)
}
```

```
bootSampleAll = boostrapTau(dParty, get_tau_cross, 1000)
```

```
# confidence interval for treatment effect when running on entire dataset
ci(tau_hat_all, var(bootSampleAll))
```

```
## [1] -0.0498380  0.4251588
```

```
# Run Lin's for each group and get confidence intervals for the treatment effect for each group
runLins = function(pid){
  data = filter(dParty, pid3 == pid)
  tau_est = get_tau_cross(data)
  bootSample = boostrapTau(data, get_tau_cross, 1000)

  ci_boot = ci(tau_est, var(bootSample))
  return (c(tau_est, ci_boot))
}

(l1 = runLins(1)) # dem
```

```
## [1]  0.1464305 -0.1822321  0.4750931
```

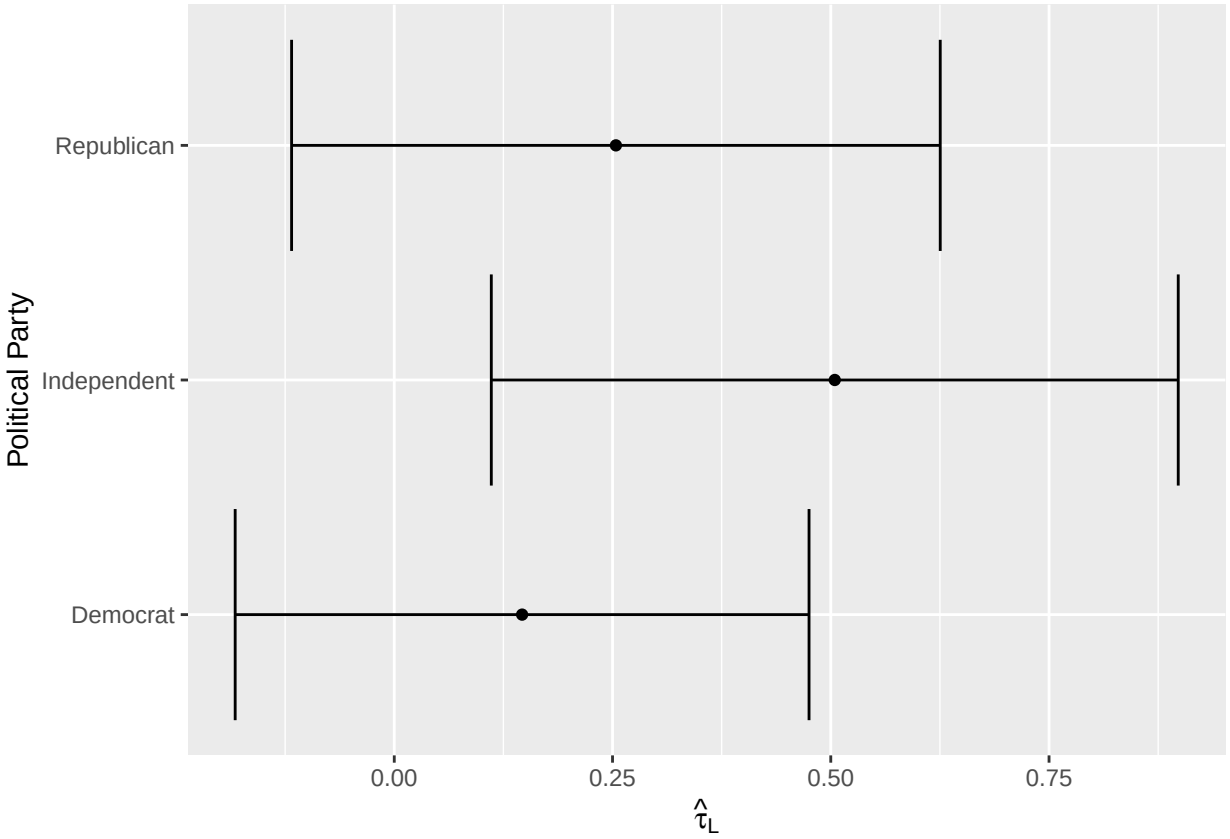
```
(l2 = runLins(2)) # ind
```

```
## [1] 0.5045623 0.1111378 0.8979869
```

```
(l3 = runLins(3)) # rep
```

```
## [1]  0.2538867 -0.1176119  0.6253853
```

```
# plot confidence intervals for treatment effect from generalized Lin's estimator, per group
confIntPlot(l1, l2, l3, "$\\hat{\\tau}_L$")
```


Republican
Independent
Democrat
Political Party
0.00
0.25
0.50
0.75
$\hat{\tau}_L$