Kaicheng Luo

Assignment 3

Stat 157
October 18, 2019

1. A predictive estimator and Lin's estimator

The two OLS regression we use are,

$$\begin{aligned} Y_i(1) &= \gamma_1 + \beta_1^T X_i(1) \\ Y_i(0) &= \gamma_0 + \beta_0^T X_i(0) \end{aligned} \tag{1}$$

For OLS properties, we have

$$\begin{aligned} \bar{Y}(1) &= \gamma_1 + \beta_1^T \bar{X}_i(1) \\ \bar{Y}(0) &= \gamma_0 + \beta_0^T \bar{X}_i(0) \end{aligned} \tag{2}$$

The predictive estimator can be written as

$$\begin{aligned} \hat{\tau}_{pre} &= \frac{1}{n}\{n_1\bar{Y}(1) + \sum_{z=0}(\gamma_1 + \beta_1^T X_i(0)) - n_0 Y\bar{(0)} - \sum_{z=1}(\gamma_0 + \beta_0^T X_i(1))\} \\ &= \frac{1}{n}\{n_0\gamma_1 - n_1\gamma_0 + n_1\bar{Y}(1) - n_0\bar{Y}(0) + n_0\beta_1^T X\bar{(0)} - n_1\beta_0^T\bar{X}(1)\} \end{aligned} \tag{3}$$

Note that we assumed mean-centered data,

$$n_0\bar{X}(0) + n_1\bar{X}(1) = n\bar{X} = 0 \tag{4}$$

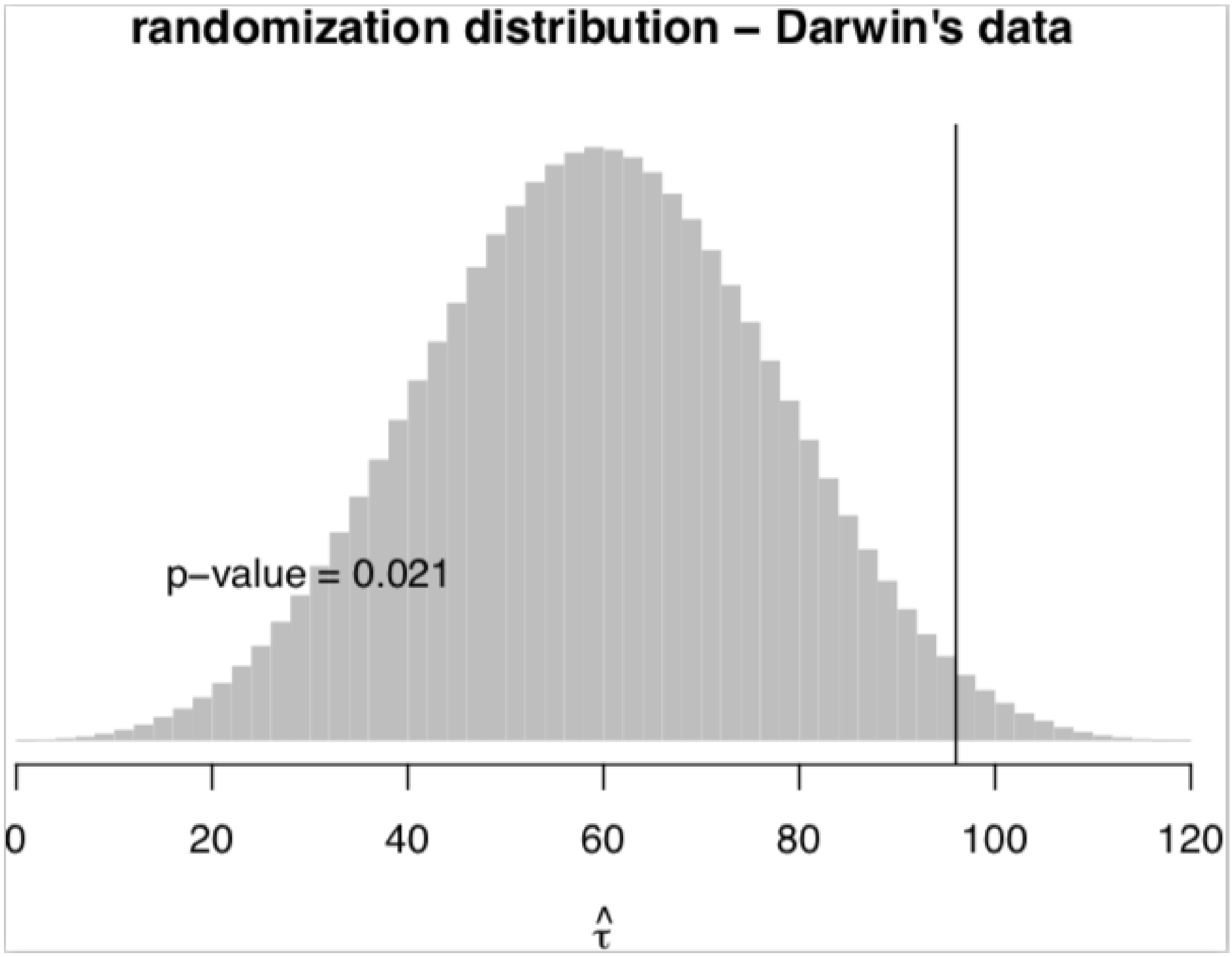
substitute $\bar{X}(z)$ by $\bar{X}(1-z)$, [3] can be further written as

$$\hat{\tau}_{pre} = \frac{1}{n}(n_0\gamma_1 + n_1\gamma_1) - \frac{1}{n}(n_1\gamma_0 + n_0\gamma_0) = \gamma_1 - \gamma_0 = \tau_L \tag{5}$$

Problem 2

```r
permute <- function(stratum, treatment){
  ptreat <- vector()
  for (i in 1:length(unique(stratum))){
    ptreat <- c(ptreat, sample(treatment[stratum == i]))
  }
  return(ptreat)
}
MC = 1000
MP_enumerate = function(i, n.pairs = 15)
{
 a = 2^((n.pairs-1):0)
 b = 2*a
 2*sapply(i-1,
          function(x)
            as.integer((x %% b)>=a)) - 1
}
## Darwin's data from Fisher's book
ytreat     = c(188, 96, 168, 176, 153,
               172, 177, 163, 146, 173,
               186, 168, 177, 184, 96)
ycontrol   = c(139, 163, 160, 160, 147,
               149, 149, 122, 132, 144,
               130, 144, 102, 124, 144)
difference = ytreat - ycontrol
n.pairs    = length(difference)
abs.diff   = abs(difference)
w.obs = wilcox.test(ytreat, ycontrol, paired = TRUE)$statistic
w.ran      = sapply(1:2^15,
                    function(x){
                      wilcox.test(ytreat*(-MP_enumerate(x)), ycontrol*(MP_enumerate(x))
                      }, simplify = TRUE)
pvalue     = mean(w.ran>=w.obs)

hist(w.ran, breaks = 50, col = "grey", border = NA,
     xlab = expression(hat(tau)),
     ylab = "", yaxt = 'n',
     main = "randomization distribution - Darwin's data")
abline(v = w.obs)
text(30, 400,
     paste("p-value = ", round(pvalue, 3), sep = ""))
```



```
# Neymanian Inference
tau_hat = mean(ytreat - ycontrol)
V_hat = sum((ytreat - ycontrol - tau_hat)^2) /n.pairs / (n.pairs-1)
print(paste("Confidence Interval = [", tau_hat - 1.96*sqrt(V_hat), ",", tau_hat + 1.96*s
```

```
## [1] "Confidence Interval = [1.83204244036545,40.0346242263012]"
```

```
## Imbens and Rubin book: matched pair data
## television program aimed at improving reading skills for children
dataxy = c(12.9, 12.0, 54.6, 60.6,
           15.1, 12.3, 56.5, 55.5,
           16.8, 17.2, 75.2, 84.8,
           15.8, 18.9, 75.6, 101.9,
           13.9, 15.3, 55.3, 70.6,
           14.5, 16.6, 59.3, 78.4,
           17.0, 16.0, 87.0, 84.2,
           15.8, 20.1, 73.7, 108.6)
```

```
dataxy = matrix(dataxy, 8, 4,  byrow = TRUE)

diffx = dataxy[, 2] - dataxy[, 1]
diffy = dataxy[, 4] - dataxy[, 3]

dataxy = cbind(dataxy, diffx, diffy)

rownames(dataxy) = 1:8
colnames(dataxy) = c("x.control", "x.treatment",
                     "y.control", "y.treatment",
                     "diffx", "diffy")
dataxy
```

```
##   x.control x.treatment y.control y.treatment diffx diffy
## 1      12.9        12.0      54.6        60.6  -0.9   6.0
## 2      15.1        12.3      56.5        55.5  -2.8  -1.0
## 3      16.8        17.2      75.2        84.8   0.4   9.6
## 4      15.8        18.9      75.6       101.9   3.1  26.3
## 5      13.9        15.3      55.3        70.6   1.4  15.3
## 6      14.5        16.6      59.3        78.4   2.1  19.1
## 7      17.0        16.0      87.0        84.2  -1.0  -2.8
## 8      15.8        20.1      73.7       108.6   4.3  34.9
```

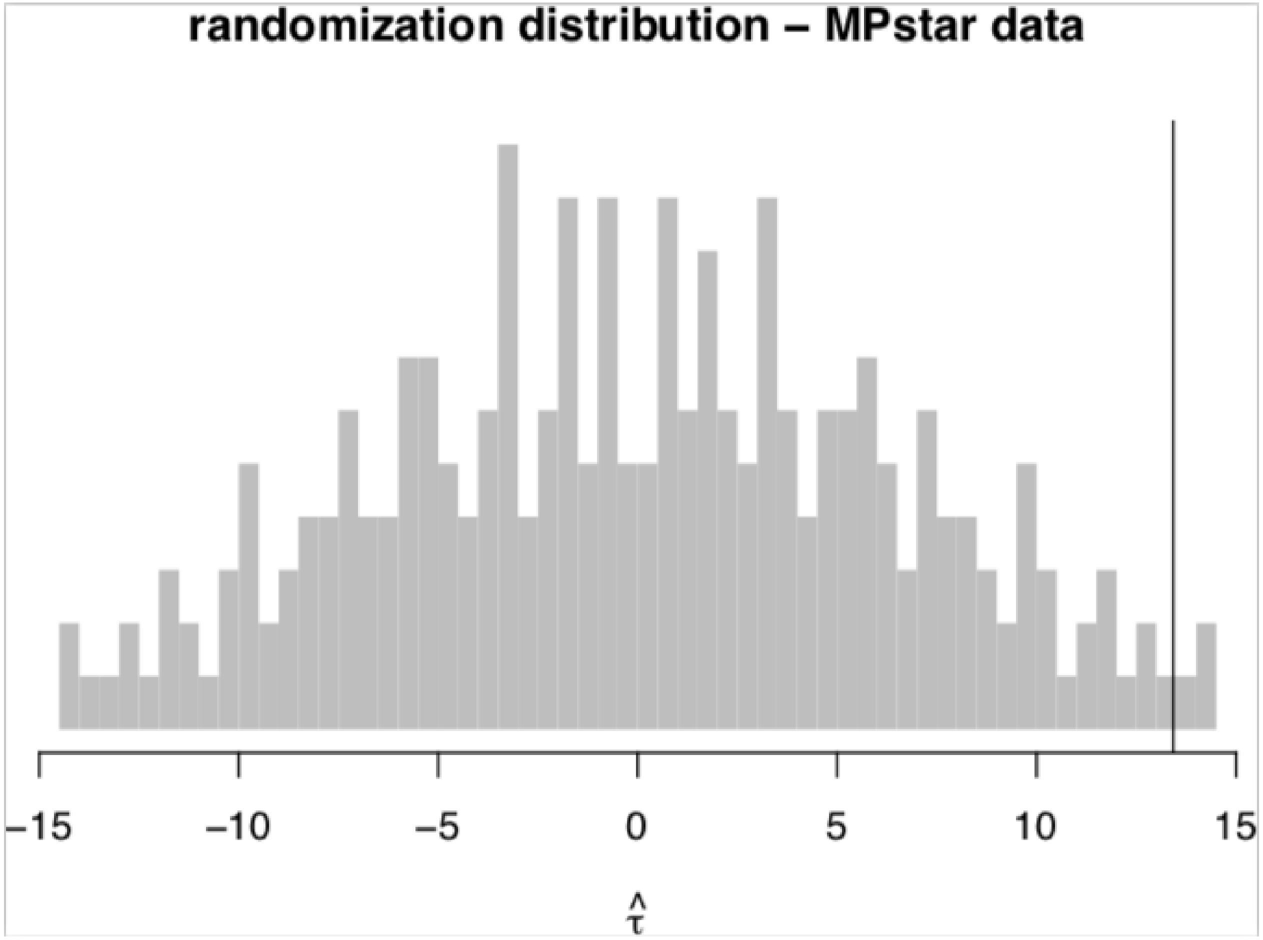
```
ytreat = dataxy[, 'y.treatment']
ycontrol = dataxy[, 'y.control']
difference = ytreat - ycontrol
n.pairs    = length(difference)
abs.diff   = abs(difference)
t.obs      = mean(difference)
t.ran      = sapply(1:2^8,
                    function(x){
                      sum(MP_enumerate(x, 8)*abs.diff)
                      })/n.pairs
w.obs = wilcox.test(ytreat, ycontrol, paired = TRUE)$statistic
w.ran      = sapply(1:2^8,
                    function(x){
                      wilcox.test(ytreat*(-MP_enumerate(x, 8)), ycontrol*(MP_enumerate(
                      }, simplify = TRUE)
pvalue     = mean(t.ran>=t.obs)
pvalue
```

```
## [1] 0.015625
```

```r
pw = mean(w.ran >= w.obs)
pw
```

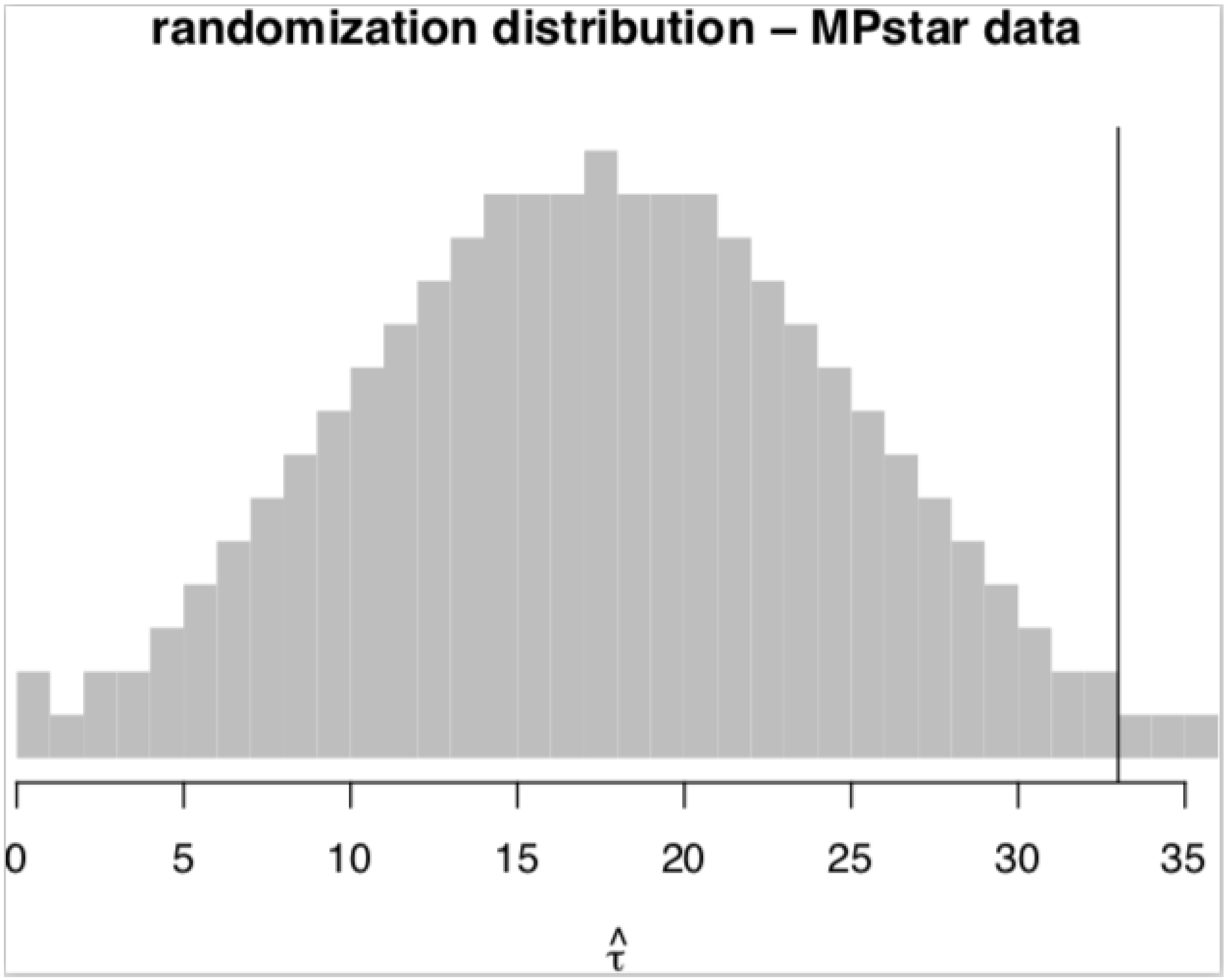
```
## [1] 0.01953125
```

```r
hist(t.ran, breaks = 50, col = "grey", border = NA,
     xlab = expression(hat(tau)),
     ylab = "", yaxt = 'n',
     main = "randomization distribution - MPstar data")
abline(v = t.obs)
text(30, 400,
     paste("p-value = ", round(pvalue, 3), sep = ""))
```



```r
hist(w.ran, breaks = 50, col = "grey", border = NA,
     xlab = expression(hat(tau)),
     ylab = "", yaxt = 'n',
     main = "randomization distribution - MPstar data")
```

```
abline(v = w.obs)
text(30, 400,
     paste("p-value = ", round(pvalue, 3), sep = ""))
```



```
# Regression Adjustment
ycontrol = lm(y.control~x.control, data = data.frame(y.control = dataxy[,'y.control'], x
ytreat = lm(y.treat~x.treat, data = data.frame(y.treat = dataxy[,'y.treatment'], x.treat
difference = ytreat - ycontrol
n.pairs    = length(difference)
abs.diff   = abs(difference)
t.obs      = mean(difference)
t.ran      = sapply(1:2^8,
                    function(x){
                      sum(MP_enumerate(x, 8)*abs.diff)
                      })/n.pairs
```

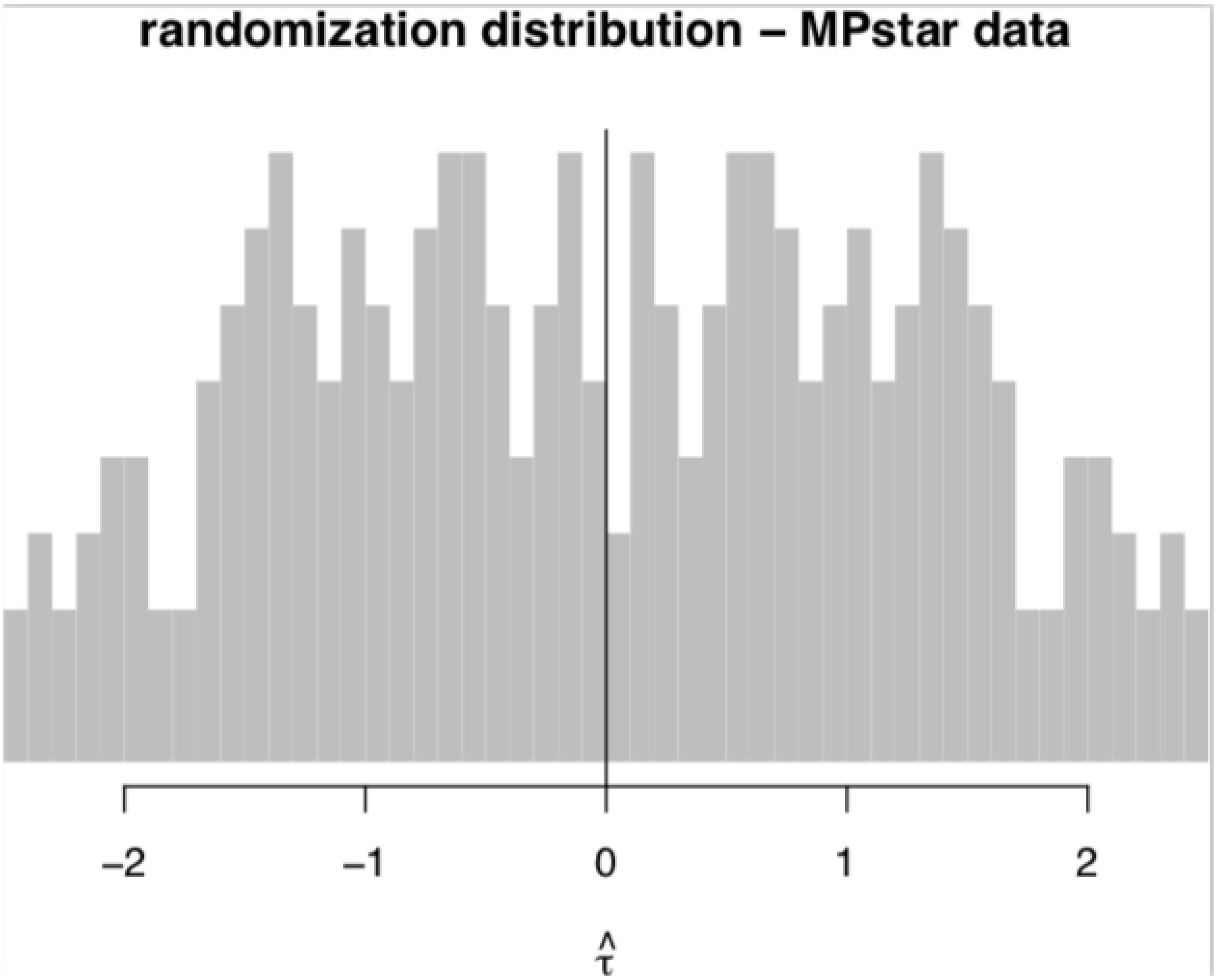
```
w.obs = wilcox.test(ytreat, ycontrol, paired = TRUE)$statistic
w.ran      = sapply(1:2^8,
                    function(x){
                      wilcox.test(ytreat*(-MP_enumerate(x, 8)), ycontrol*(MP_enumerate(
                      }, simplify = TRUE)
pvalue     = mean(t.ran>=t.obs)
pvalue
```

```
## [1] 0.5
```

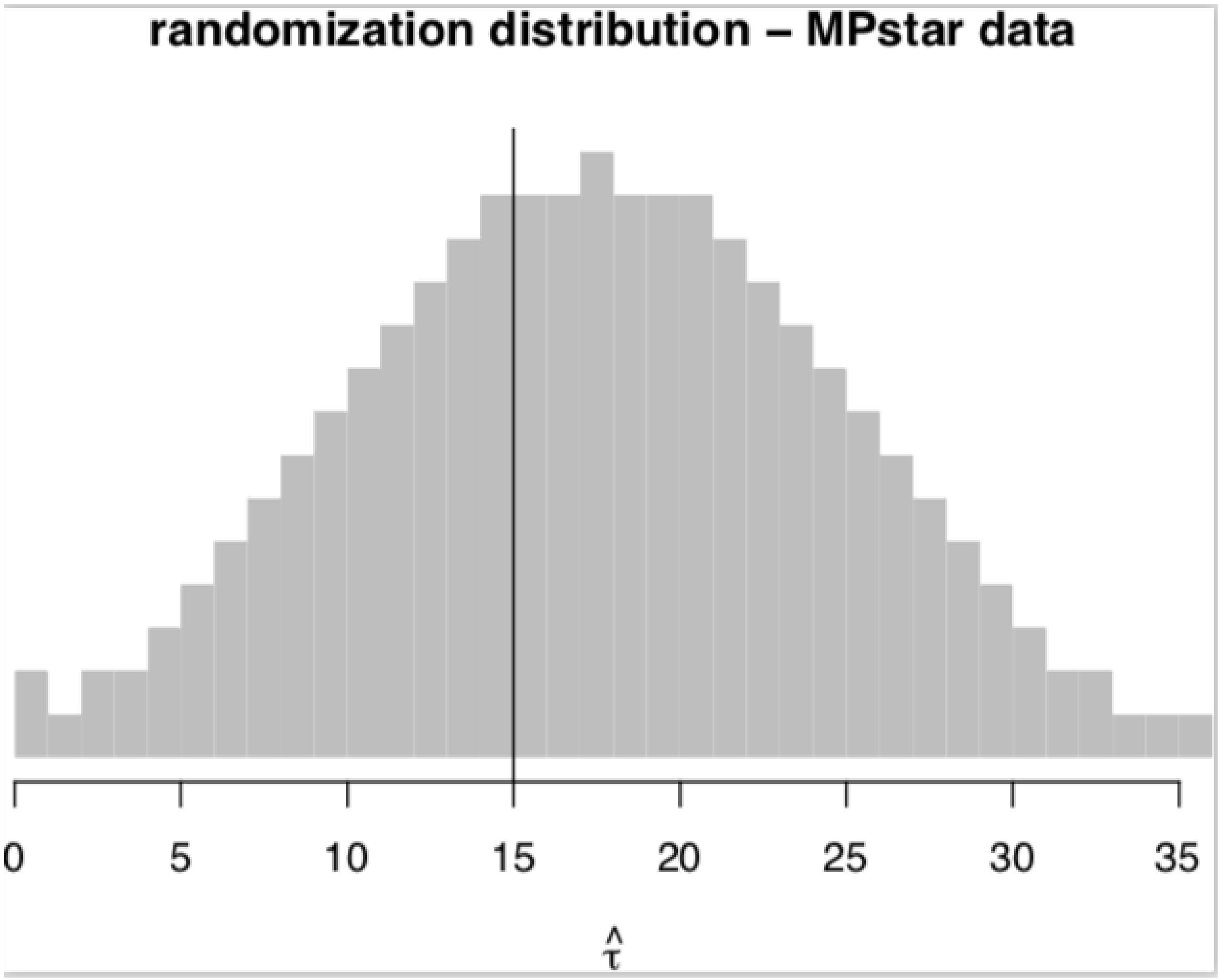
```
pw = mean(w.ran >= w.obs)
pw
```

```
## [1] 0.6796875
```

```
hist(t.ran, breaks = 50, col = "grey", border = NA,
     xlab = expression(hat(tau)),
     ylab = "", yaxt = 'n',
     main = "randomization distribution - MPstar data")
abline(v = t.obs)
text(30, 400,
     paste("p-value = ", round(pvalue, 3), sep = ""))
```



```r
hist(w.ran, breaks = 50, col = "grey", border = NA,
     xlab = expression(hat(tau)),
     ylab = "", yaxt = 'n',
     main = "randomization distribution - MPstar data")
abline(v = w.obs)
text(30, 400,
     paste("p-value = ", round(pvalue, 3), sep = ""))
```



```
# Data Input
dataxy = c(.046, 0, .091, .185, .036, .051, 0, .047,
           .054, .094, .184, .034, .05, .108, .11, .095,
           .114, 0, .056, .075, .098, .054, .030, .068,
           .148, .162, .082, .075, .134, .39, .339, .458,
           .152, .105, .083, .129, .145, .077, .579, .167,
           .188, .214, .375, .545, .179, .165, .483, .444,
           .193, .771, .328, .583, .189, .186, .168, .368,
           .197, .350, .0, .383, .2, .071, .667, .429,
           .213, .176, .164, .172, .209, .165, .092, .151,
           .211, .667, .25, .617, .219, .25, .5, .35,
           .219, .153, .185, .219, .224, .363, .372, .342,
           .255, .226, .213, .327, .257, .098, .107, .095,
           .261, .071, .0, .435, .263, .441, .448, .435,
           .286, .161, .126, .181, .285, .389, .353, .309)
```

```r
data <- data.frame(matrix(dataxy, ncol = 4))
data <- data %>%
  rename(Y1 = X3, Y2 = X4) %>%
  mutate(Y = Y1 + Y2)
```

```r
# FRT with raw data
data <- data %>%
  mutate(index = 1:28) %>%
  mutate(pair = (index+1) %/% 2) %>%
  mutate(z = (index+1) %% 2) %>%
  select(-index, -Y1, -Y2)
stat_Pair <- function(pair, treatment, y){
  # Assume in our case that the stratum in arranged and indexed.
  # If not, then re-code it to an index.
  number = length(unique(pair))
  tau = 0
  temptau = rep(0, number)
  wil = 0
  sign = 0
  # Calculate the three statistics as defined
  for (i in 1:number){
    tempy = y[pair == i]
    tempt = treatment[pair == i]
    tau = tau + (tempy[tempt == 1] - tempy[tempt == 0])/number
    temptau[i] = tempy[tempt == 1] - tempy[tempt == 0]
    sign = sign + ifelse(tempy[tempt == 1] > tempy[tempt == 0], 1, 0)
  }
  Wrank = rank(temptau)
  for (i in 1:number){
    tempy = y[pair == i]
    tempt = treatment[pair == i]
    wil = wil + ifelse(tempy[tempt == 1] > tempy[tempt == 0], 1, 0)*Wrank[i]
  }
  return(c(taus = tau, wilcoxon = wil, alignedRank = sign))
}
obsValue <- stat_Pair(data$pair, data$z, data$Y)
```

```r
ext = rep(0,3)
for (i in 1:MC){
  rvalue = stat_Pair(data$pair, permute(data$pair, data$z), data$Y)
  for (j in 1:3){
    if (abs(obsValue[j]) > abs(rvalue[j])){
      ext[j] = ext[j]+1
    }
  }
}
ext/MC
```

```
## [1] 0.742 0.226 0.226
```

```
# FRT with covariate-adjusted data
dataxy = c(.046, 0, .091, .185, .036, .051, 0, .047,
           .054, .094, .184, .034, .05, .108, .11, .095,
           .114, 0, .056, .075, .098, .054, .030, .068,
           .148, .162, .082, .075, .134, .39, .339, .458,
           .152, .105, .083, .129, .145, .077, .579, .167,
           .188, .214, .375, .545, .179, .165, .483, .444,
           .193, .771, .328, .583, .189, .186, .168, .368,
           .197, .350, .0, .383, .2, .071, .667, .429,
           .213, .176, .164, .172, .209, .165, .092, .151,
           .211, .667, .25, .617, .219, .25, .5, .35,
           .219, .153, .185, .219, .224, .363, .372, .342,
           .255, .226, .213, .327, .257, .098, .107, .095,
           .261, .071, .0, .435, .263, .441, .448, .435,
           .286, .161, .126, .181, .285, .389, .353, .309)
data <- data.frame(matrix(dataxy, ncol = 4))
data <- data %>%
  rename(Y1 = X3, Y2 = X4) %>%
  mutate(Y = lm((Y1+Y2)~X1+X2)$residuals)
```

```
data <- data %>%
  mutate(index = 1:28) %>%
  mutate(pair = (index+1) %/% 2) %>%
  mutate(z = (index+1) %% 2) %>%
  select(-index, -Y1, -Y2)
stat_Pair <- function(pair, treatment, y){
  # Assume in our case that the stratum in arranged and indexed.
  # If not, then re-code it to an index.
  number = length(unique(pair))
  tau = 0
  temptau = rep(0, number)
  wil = 0
  sign = 0
  # Calculate the three statistics as defined
  for (i in 1:number){
    tempy = y[pair == i]
    tempt = treatment[pair == i]
    tau = tau + (tempy[tempt == 1] - tempy[tempt == 0])/number
    temptau[i] = tempy[tempt == 1] - tempy[tempt == 0]
    sign = sign + ifelse(tempy[tempt == 1] > tempy[tempt == 0], 1, 0)
  }
  Wrank = rank(temptau)
  for (i in 1:number){
    tempy = y[pair == i]
    tempt = treatment[pair == i]
    wil = wil + ifelse(tempy[tempt == 1] > tempy[tempt == 0], 1, 0)*Wrank[i]
  }
  return(c(taus = tau, wilcoxon = wil, alignedRank = sign))
}
obsValue <- stat_Pair(data$pair, data$z, data$Y)
```

```
ext = rep(0,3)
for (i in 1:MC){
  rvalue = stat_Pair(data$pair, permute(data$pair, data$z), data$Y)
  for (j in 1:3){
    if (abs(obsValue[j]) > abs(rvalue[j])){
      ext[j] = ext[j]+1
    }
  }
}
```

```
# Neyman Inference with raw data
number = 14
temptau = rep(0, number)
for (i in 1:number){
  tempy = data$Y[data$pair == i]
  tempt = data$z[data$pair == i]
  temptau[i] = tempy[tempt == 1] - tempy[tempt == 0]
}
print(paste("The point estimator for Tau_pair is ", mean(temptau), sep = ""))
```

```
## [1] "The point estimator for Tau_pair is 0.0720902807264858"
```

```
print(paste("The variance estimator for Tau_pair is ", sum((temptau - mean(temptau))^2)/
```

```
## [1] "The variance estimator for Tau_pair is 0.0050435225427803"
```

```
print(paste("The Confidence Interval for Tau_pair is [", mean(temptau) - 1.96*sum((tempt
```

```
## [1] "The Confidence Interval for Tau_pair is [0.0622049765426364,0.0819755849103352]"
```

```
# Neyman Inference with covariate-adjusted data
tempx = matrix(0, nrow = 14, ncol = 2)
for (i in 1:number){
  tempX1 = data$X1[data$pair == i]
  tempX2 = data$X2[data$pair == i]
  tempt = data$z[data$pair == i]
  tempx[i,1] = tempX1[tempt == 1] - tempX2[tempt == 0]
  tempx[i,2] = tempX2[tempt == 1] - tempX2[tempt == 0]
}

tau = lm(temptau~tempx)$coef[1]
v = summary(lm(temptau~tempx))$coef[1,2]
print(paste("The point estimator for Tau_pair is ", tau, sep = ""))
```

```
## [1] "The point estimator for Tau_pair is 0.00307026466222173"
```

```
print(paste("The variance estimator for Tau_pair is ", v,sep = ""))
```

```
## [1] "The variance estimator for Tau_pair is 0.13923301432281"
```

```
print(paste("The Confidence Interval for Tau_pair is [", mean(temptau) - 1.96*v, ",", me
```

```
## [1] "The Confidence Interval for Tau_pair is [-0.200806427346222,0.344986988799194]"
```

Problem 3

First prove the lemma that

$$\mathbb{E}(\sum_{i=1}^{n} x_i y_i) = cov(x_i, y_i) + \sum_{i=1}^{n} \bar{x}_i \bar{y}_i \tag{6}$$

Proof :

$$\begin{aligned} RHS &= \mathbb{E}[\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) + \sum_{i=1}^{n} \bar{x}_i \bar{y}_i] = \mathbb{E}[\sum_{i=1}^{n} x_i y_i - \bar{y}\sum_{i=1}^{n} X_i - \bar{x}\sum_{i=1}^{n} y_i + \bar{x}\bar{y}] + \sum_{i=1}^{n} \bar{x}_i \bar{y}_i \\ &= \mathbb{E}[\sum_{i=1}^{n} x_i y_i] \end{aligned} \tag{7}$$

Multiply both side by $n(n-1)$,

$$\begin{aligned} n(n-1)\mathbb{E}(\hat{\theta}) &= \mathbb{E}[\sum_{i=1}^{n}(\hat{\tau}_{xi}-\hat{\tau}_x)(\hat{\tau}_i-\hat{\tau})] \\ &= \mathbb{E}[\sum_{i=1}^{n}\hat{\tau}_{xi}\hat{\tau}_i - n\hat{\tau}_x\hat{\tau}] \\ &= cov(\hat{\tau}_{xi},\hat{\tau}_i) + \mathbb{E}(n\hat{\tau}_x\hat{\tau}) - ncov(\hat{\tau},\hat{\tau}_x) - \mathbb{E}(n\hat{\tau}_x\hat{\tau}) \\ &= n(n-1)cov(\hat{\tau},\hat{\tau}_x) \end{aligned} \tag{8}$$

So $\hat{\theta}$ gives an unbiased estimate of the covariance.

Problem 4

```r
# Stratums that have only treatment / control units
data("homocyst")
temp <- homocyst %>%
  group_by(st) %>%
  filter(z == 1) %>% count()
homocyst <- homocyst %>%
  mutate(yestreat = ifelse(st %in% temp$st, 1, 0))
temp <- homocyst %>%
  group_by(st) %>%
  filter(z == 0) %>% count()
homocyst <- homocyst %>%
  mutate(yescon = ifelse(st %in% temp$st, 1, 0)) %>%
  mutate(yes = yescon + yestreat)
homocyst %>%
  filter(yes != 2) %>%
  arrange(by = st) %>%
  distinct(st)
```

```
##     st
## 1    4
## 2    6
## 3   17
## 4   29
## 5   46
## 6   47
## 7   48
## 8   51
## 9   53
## 10  54
## 11  57
## 12  69
## 13  76
## 14  78
## 15  94
## 16 100
## 17 105
## 18 106
```

```
# The proportion of the units to be dropped is 5.01%
homocyst %>%
  filter(yes!=2) %>%
  nrow() / nrow(homocyst)
```

```
## [1] 0.05010101
```

```
homocyst <- homocyst %>%
  filter(yes==2)
stnum = 1
data <- homocyst %>%
  mutate(stratum = 0) %>%
  arrange(by = st)
for (i in 2:nrow(data)){
  if (data$st[i] == data$st[i-1]){
    data$stratum[i] = stnum
  }else{
    stnum = stnum + 1
    data$stratum[i] = stnum
  }
}
data$stratum[1] = 1
```

```
# Stratified RE
stat_SRE <- function(stratum, treatment, y){
  # Assume in our case that the stratum in arranged and indexed.
  # If not, then re-code it to an index.
  number = length(unique(stratum))
  tau = 0
  wil = 0
  r = 0
  # Calculate the three statistics as defined
  for (i in 1:number){
    tempy = y[stratum == i]
    tempt = treatment[stratum == i]
    n = length(tempy)
    pi = n/length(y)
    tau = tau + pi*(mean(tempy[tempt == 1] - mean(tempy[tempt == 0])))
    wil = wil + wilcox.test(tempy[tempt == 1], tempy[tempt == 0])$statistic / (n+1)
    tempy = tempy - mean(tempy)
  }
  y <- rank(y)
  for (i in 1:length(y)){
    if (treatment[i] == 1){
      r = r + y[i]
    }
  }
  return(c(taus = tau, wilcoxon = wil, alignedRank = r))
}
# Here we obtain the obs. values
obsValue <- stat_SRE(data$stratum, data$z, data$homocysteine)
# This is a function for blocked permutation
permute <- function(stratum, treatment){
  ptreat <- vector()
  for (i in 1:length(unique(stratum))){
    ptreat <- c(ptreat, sample(treatment[stratum == i]))
  }
  return(ptreat)
}

MC = 2000
extreme = rep(0,3)
for (i in 1:MC){
  mcStat = stat_SRE(data$stratum, permute(data$stratum, data$z), data$homocysteine)
  for (j in 1:3){
    if (abs(mcStat[j]) > abs(obsValue[j])){
      extreme[j] = extreme[j] + 1
    }
  }
}
# Tidy display of our result
display <- data.frame("Taus" = extreme[1]/MC, "V" = extreme[2]/MC, "Aligned Rank" = extr
display
```

```
##   Taus V Aligned.Rank
## 1    0 0            0
```

```
# p-value is actually smaller than 1/2000
```

```
# Neymanian Inference
print(c ("The point estimator is", obsValue[1]))
```

```
##                                                taus
## "The point estimator is"       "1.67069163556073"
```

```
var_neyman <- function(stratum, treatment, y){
  V = 0
  for(i in 1:length(unique(stratum))){
    tempy = y[stratum == i]
    tempt = treatment[stratum == i]
    n = length(tempy)
    y0 = tempy[tempt == 0]
    y1 = tempy[tempt == 1]
    # We have to deal with the situation where we do not have enough data to calculate
    if (length(y0) == 1){
      sd0 = 0
    }else{
      sd0 = (sd(y0))^2
    }
    if (length(y1) == 1){
      sd1 = 0
    }else{
      sd1 = (sd(y1))^2
    }
    V = V + (length(n)/length(y))^2 * (sd0/length(y0) + sd1/length(y1))
  }
  return(V)
}
print(paste("The variance estimator is ", var_neyman(data$stratum, data$z, data$homocyst
```

```
## [1] "The variance estimator is 0.000180035179461383"
```

```
print(paste("The confidence interval is [", obsValue[1] - 1.96 * sqrt(var_neyman(data$st
```

```
## [1] "The confidence interval is [1.64439290659108,1.69699036453037]"
```

```
# Baseline OLS estimator
data <- data %>%
  select(-st, -SEQN, -stf, -yestreat, -yescon, -yes, -stratum)
```

```r
model <- lm(homocysteine ~ ., data = data)
stargazer(model)
```

Table 1

	Dependent variable:
	homocysteine
z	1.378$***$
	(0.253)
female	$-$1.477$***$
	(0.209)
age3	1.806$***$
	(0.130)
ed3	$-$0.090
	(0.137)
bmi3	$-$0.090
	(0.136)
pov2	$-$0.331
	(0.230)
Constant	6.034$***$
	(0.471)
Observations	2,351
R^2	0.116
Adjusted R^2	0.114
Residual Std. Error	4.915 (df = 2344)
F Statistic	51.161$***$ (df = 6; 2344)
Note:	$*$$p$ < 0.1; $**$$p$ < 0.05; $***$$p$ < 0.01

```r
# Lin's estimator
Lin <- lm(homocysteine ~ . + z:., data = data)
print(paste("The point estimator (By Lin) is ", Lin$coefficients['z'], sep = ""))
```

```
## [1] "The point estimator (By Lin) is 2.16026928057566"
```

```
print(paste("The variance estimator (By Lin) is ", hccm(Lin)['z','z'], sep = ""))
```

```
## [1] "The variance estimator (By Lin) is 2.61230710334533"
```

Lin's estimator is actually most credible, because for FRT, we're dealing with some stratum that has only one observation, dropping them leads to bias, and somehow the stratums are designed arbitrarily. Concerning baseline OLS regression, we have a biased estimate but not necessarily lowering the bias. Lin's estimator solves the problem.

Problem 5

$$\begin{aligned}\mathbb{E}\{\sum_{i=1}^n \frac{z_i}{e(x_i)}\} &= \mathbb{E}\{\mathbb{E}\{\sum_{i=1}^n \frac{z_i}{e(x_i)}|x_i\}\} \\ &= \mathbb{E}\{\mathbb{E}\{\sum_{i=1}^n \frac{e(x_i)}{e(x_i)}|x_i\}\} \\ &= n\end{aligned} \tag{9}$$

$$\begin{aligned}\mathbb{E}\{\sum_{i=1}^n \frac{1-z_i}{1-e(x_i)}\} &= \mathbb{E}\{\mathbb{E}\{\sum_{i=1}^n \frac{1-z_i}{1-e(x_i)}|x_i\}\} \\ &= \mathbb{E}\{\mathbb{E}\{\sum_{i=1}^n \frac{1-e(x_i)}{1-e(x_i)}|x_i\}\} \\ &= n\end{aligned} \tag{10}$$

Note that $\mathbb{E}(z_i|x_i) = e(x_i)$, which is fixed given x_i.

Problem 6

```
data <- read.csv("Rubin_data(1983).csv")
# Neymanian Inference
subtau <- data$improved[data$treatment == 1] - data$improved[data$treatment == 0]
subpi <- data %>% group_by(subclass) %>% summarise(n = sum(patients)) %>% pull(n) / sum
subvar <- data$se[data$treatment == 1]^2/data$patients[data$treatment == 1] + data$se[da
print(paste("The point estimator is ", sum(subtau*subpi), sep = ""))
```

```
## [1] "The point estimator is 0.312"
```

```
print(paste("The variance estimator is ", sum(subpi^2*subvar), sep = ""))
```

```
## [1] "The variance estimator is 2.2072913983781e-05"
```

```r
print(paste("The confidence interval is [", sum(subtau*subpi) - 1.96*sum(subpi^2*subvar)
```

```
## [1] "The confidence interval is [0.311956737088592,0.312043262911408]"
```