Assn 2b: Randomized Experiment Simulation Homework

Kenny Mai

Introduction

Randomized experiments are called the "gold standard" due to their ability to unbiasedly answer causal questions. This is achieved by creating two (or more) groups that are identical to each other on average, in terms of the distribution of all (pre-treatment) variables. So, if each group receives a different treatment and the groups have different outcomes, we can safely attribute these differences due only to the systematic difference between groups: the treatment.

In a randomized experiment, units are assigned to treatments using a known probabilistic rule. Each unit has nonzero probability of being allocated each treatment group. In class, we discussed two major types of randomized experiments that differ based on different assignment rules: **completely randomized assignment** and **randomized block assignment**.

Question 1

Recall that in Assignment 2a we created a simulated dataset that could have manifested as a result of a completely randomized experiment. In that assignment, we asked about the difference between estimating the Sample Average Treatment Effect (SATE) by using the difference in means versus using linear regression with pretest score as a covariate. However we only looked at one realized dataset so couldn't make more general comments about bias and efficiency of these estimators. In this exercise, we will further explore these properties of these two different approaches to estimating our ATEs through simulation. For this question you will need to use the function dgp1.fun from assignment 2a.

```
# Loading needed libraries
library(tidyverse)
```

```
## -- Attaching packages --- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
dgp1.fun <- function(N,coef,seed){
  set.seed(seed)
  #Create pre-treatment test scores for everyone
  pretest <- rnorm(n=N, mean = 65, sd = 3)
  #Create potential outcome where tau = 5
  y0 <- 10 + coef * pretest + 0 + rnorm(n = N, mean = 0, sd = 1)
  y1 <- 10 + coef * pretest + 5 + rnorm(n = N, mean = 0, sd = 1)
  dat<-data.frame(pretest=pretest,y0=y0,y1=y1)
return(dat)
```

```
}
```

(a) Start by drawing a sample of size 100 using this function, again setting the coefficient on the pretest to be 1.1 and the seed to be 1234.

```
# Draw a sample
samp1 = dgp1.fun(100,1.1,1234)
samp1
```

```
##      pretest       y0       y1
## 1   61.37880 77.93121 83.00191
## 2   65.83229 81.94080 88.11229
## 3   68.25332 85.14465 90.26417
## 4   57.96291 73.25672 79.45993
## 5   66.28737 82.09011 88.22779
## 6   66.51817 83.33697 88.93045
## 7   63.27578 78.70709 86.44582
## 8   63.36010 79.86430 85.80848
## 9   63.30664 79.99228 84.66997
## 10  62.32989 78.51077 82.44843
## 11  63.56842 79.72933 85.34332
## 12  62.00484 77.55625 82.80509
## 13  62.67124 77.82859 85.43186
## 14  65.19338 82.56199 85.10563
## 15  67.87848 84.68869 89.25058
## 16  64.66914 81.96720 86.55807
## 17  63.46697 78.56938 84.66193
## 18  62.26641 78.66208 82.88690
## 19  62.48848 79.41050 83.43261
## 20  72.24751 89.44598 95.10179
## 21  65.40226 81.75110 87.83766
## 22  63.52794 79.09883 85.54095
## 23  63.67836 82.10435 87.31968
## 24  66.37877 83.76715 89.19014
## 25  62.91884 81.03493 84.49843
## 26  60.65539 76.80098 81.06115
## 27  66.72427 82.76528 91.31583
## 28  61.92903 76.60865 83.79935
## 29  64.95459 80.81394 85.76572
## 30  62.19215 78.63767 83.59786
## 31  68.30689 86.15127 89.81319
## 32  63.57322 80.18329 84.65584
## 33  62.87168 77.98690 83.22534
## 34  63.49623 80.51456 84.96269
## 35  60.11272 74.47389 81.44315
## 36  61.49714 77.28100 81.56931
## 37  58.45988 73.98975 76.07272
## 38  60.97702 75.12648 81.81985
## 39  64.11712 81.44889 85.55835
## 40  63.60231 79.33967 85.55681
## 41  69.34849 85.94930 91.34247
## 42  61.79407 79.36863 83.38688
## 43  62.43391 79.31397 82.57952
## 44  64.15813 80.46551 86.28512
## 45  62.01698 78.73244 83.93757
```

```
## 46  62.09446 78.70317 83.55555
## 47  61.67805 79.50871 84.20312
## 48  61.24404 77.64434 82.77292
## 49  63.42852 80.27764 85.03573
## 50  63.50945 80.20795 85.12844
## 51  59.58191 75.16286 80.97703
## 52  63.25377 79.67677 85.63927
## 53  61.67333 79.47941 83.29285
## 54  61.95511 77.27503 83.81382
## 55  64.51307 81.08614 84.82801
## 56  66.68917 84.72021 87.98759
## 57  69.94345 86.70318 93.41477
## 58  62.67994 77.89455 82.72403
## 59  69.81773 85.92972 92.05757
## 60  61.52657 77.28910 83.08423
## 61  66.96977 82.81939 89.64255
## 62  72.64697 89.65103 94.56279
## 63  64.89572 80.97087 86.54392
## 64  62.99110 79.10716 82.52695
## 65  64.97719 81.88196 86.81350
## 66  70.33125 87.98901 91.69781
## 67  61.58418 79.42080 82.50395
## 68  69.10348 85.94514 89.82606
## 69  68.98869 85.56672 91.27250
## 70  66.00942 84.08137 88.27694
## 71  65.02068 83.22708 86.21813
## 72  63.63359 80.04020 86.82196
## 73  63.90043 79.95781 85.96103
## 74  66.94486 81.81711 89.58798
## 75  71.21081 89.74316 95.38130
## 76  64.53980 80.15620 85.34267
## 77  60.82790 75.78692 82.71931
## 78  62.82925 82.15595 85.09876
## 79  65.77479 82.58729 87.34609
## 80  64.04882 80.42045 85.77276
## 81  64.46663 78.18107 84.90147
## 82  64.49002 80.83923 86.40919
## 83  60.88309 77.94744 81.27043
## 84  64.47864 81.34037 86.74019
## 85  67.55070 85.21809 88.49434
## 86  67.09283 85.78584 89.12151
## 87  66.64999 84.48410 87.46847
## 88  63.79180 79.66225 84.92522
## 89  64.42522 81.57192 84.31488
## 90  61.41642 77.35964 82.68649
## 91  64.84052 80.78651 87.31002
## 92  65.76559 79.48639 87.52539
## 93  70.11789 86.34003 90.36345
## 94  68.00454 85.29281 89.18446
## 95  63.51325 82.03261 86.52062
## 96  66.06665 83.17401 89.48312
## 97  61.59618 78.37600 81.58076
## 98  67.63461 83.43217 89.03137
## 99  67.91875 84.87328 90.06425
```

```
## 100 71.36335 86.42145 93.81884
```

(b) We will now investigate the properties of two estimators of the SATE.

- difference in means
- linear regression estimate of the treatment effect using the pretest score as a covariate

For now we will only consider the variability in estimates that would manifest as a result of the randomness in who is assigned to receive the treatment (this is sometimes referred to as "randomization based inference"). Since we are in Statistics God mode we can see how the observed outcomes and estimates would change across a distribution of possible treatment assignments. We simulate this by repeatedly drawing a new vector of treatment assignments and then for each new dataset calculating estimates using our two estimators above. We will use these estimates to create a "randomization distribution" (similar to a sampling distribution) for these two different estimators for the SATE. Obtain 10,000 draws from this distribution. [Hint: Note that the only thing that will be different in each new dataset is the treatment and observed outcome; the covariate values and potential outcomes will remain the same!]

```r
# Define sample size
n = 100
# Define number of draws
N = 10000
# Initialize vector for difference in means
dif.means = rep(NA,N)
# Initialize vector ofr linear regression estimate of treatment effect
lin.reg = rep(NA,N)

# Begin look for N iterations
for (i in 1:N) {
  # Assign treatment
  treatment=rbinom(n,1,0.5)
  # Turn off God-mode; only one value for y depending on treatment assignment
  y<-ifelse(treatment==1, samp1$y1, samp1$y0)
  # Create new data frame for non-God-mode
  dat1 <- data.frame(pretest=samp1$pretest,treatment=treatment, y=y)
  # Calculate difference in means, place it into results vector dif.means
  dif.means[i] = mean(dat1$y[treatment==1]) - mean(dat1$y[treatment==0])
  # Calculate coefficient for treatment through linear regression, place into lin.reg
  lin.reg[i] = summary(lm(y ~ pretest + treatment, dat1))$coefficient[3]
}
# Quick sanity check
head(dif.means)
```

```
## [1] 5.395828 5.504861 5.222786 4.559875 5.273438 5.466191
```

```r
head(lin.reg)
```

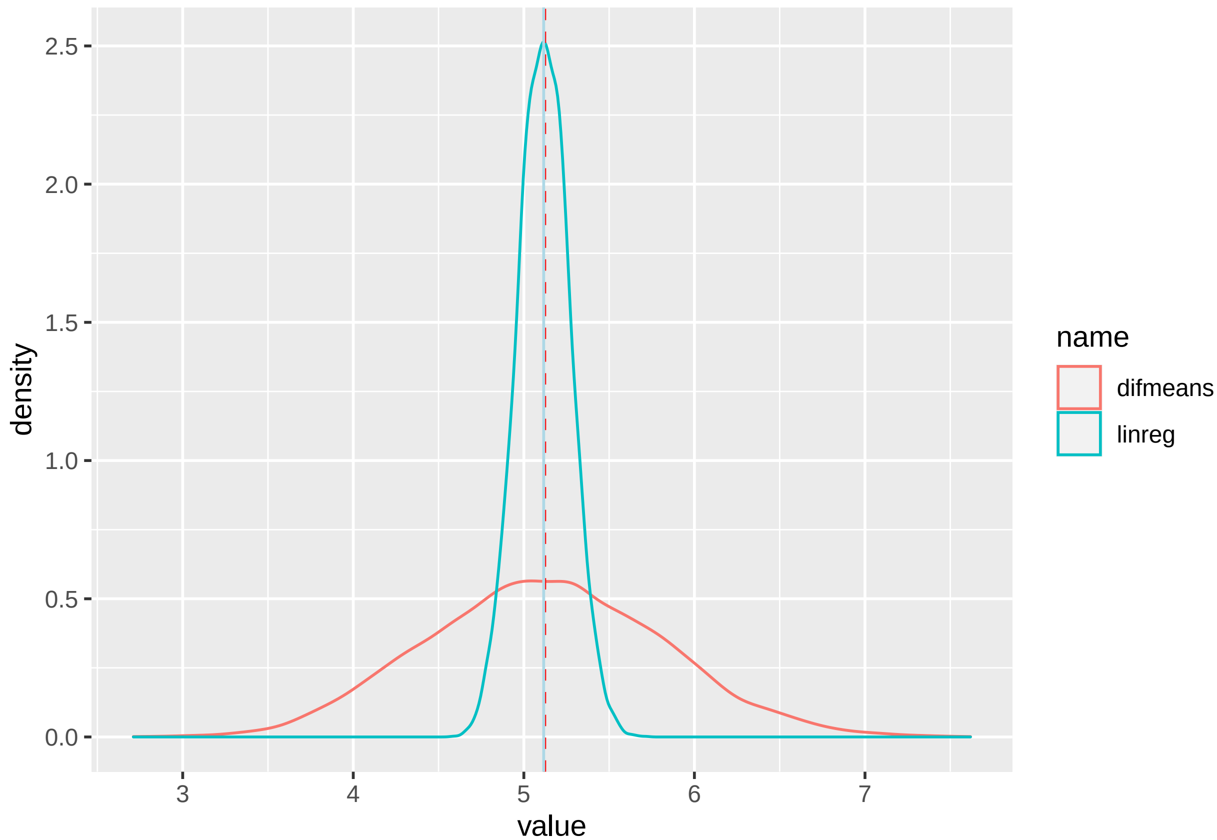
```
## [1] 5.278322 5.039244 5.021336 5.010137 4.985367 5.014435
```

```r
# Create data from to store them in
df1 = data.frame(difmeans=dif.means, linreg=lin.reg)
```

(b) Plot the (Monte Carlo estimate of the) randomization distribution for each of the two estimators: difference in means and regression. Either overlay the plots (with different colors for each) or make sure the xlim on both plots is the same. Also add vertical lines (using different colors) for the SATE and the mean of the randomizaton distribution.

```r
# Using tidyverse and ggplot to graph the difference in means and linear regression draws
# Pivot the dataframe to stack all values in one column
```

```r
df1 %>% pivot_longer(1:2) %>%
  # Call ggplot and color by name
  ggplot(aes(value,color=name)) +
    # Graph density plots of both dif.means and lin.reg
    geom_density() +
    geom_vline(xintercept = mean(df1$difmeans),color="red",linetype="dashed") +
    geom_vline(xintercept = mean(df1$linreg),color="light blue")
```



(c) Calculate the bias and efficiency of each of these two methods and compare them.

```r
# Calculate SATE from God-mode
sate1 = mean(samp1$y1 - samp1$y0)
# Bias of difference in means
mean(dif.means)-sate1
```

```
## [1] 0.009062166
```

```r
# Bias in linear regression
mean(lin.reg)-sate1
```

```
## [1] 0.00299457
```

```r
# Efficiency of difference in means
var(dif.means)
```

```
## [1] 0.4828071
```

```r
# Efficiency of linear regression
var(lin.reg)
```

```
## [1] 0.02307917
```

Comparing the bias of the difference in means and linear regression, we can see that both are identical and approximate to zero, and this makes them unbiased estimators However, comparing efficiency, we can see that using linear regression is more efficient. This can be visually seen in the graph above, as difference in means has much more variance than linear regression.

(d) Re-run the simulation with a small coefficient (even 0) for the pretest covariate. Does the small coefficient lead to a different bias and efficiency estimate compared to when the coefficient for pretest was at **1.1** from before?

```
# Draw new sample, with pretest covariate 0.1 instead of 1.1
samp2 = dgp1.fun(100,0.1,1234)
# Define sample size
n = 100
# Define number of draws
N = 10000
# Initialize vector for difference in means
dif.means = rep(NA,N)
# Initialize vector ofr linear regression estimate of treatment effect
lin.reg = rep(NA,N)

# Begin look for N iterations
for (i in 1:N) {
  # Assign treatment
  treatment=rbinom(n,1,0.5)
  # Turn off God-mode; only one value for y depending on treatment assignment
  y<-ifelse(treatment==1, samp2$y1, samp2$y0)
  # Create new data frame for non-God-mode
  dat1 <- data.frame(pretest=samp2$pretest,treatment=treatment, y=y)
  # Calculate difference in means, place it into results vector dif.means
  dif.means[i] = mean(dat1$y[treatment==1]) - mean(dat1$y[treatment==0])
  # Calculate coefficient for treatment through linear regression, place into lin.reg
  lin.reg[i] = summary(lm(y ~ pretest + treatment, dat1))$coefficient[3]
}
# Quick sanity check
head(dif.means)
```

```
## [1] 5.288299 5.070879 5.043022 4.961942 5.021635 5.056689
```

```
head(lin.reg)
```

```
## [1] 5.278322 5.039244 5.021336 5.010137 4.985367 5.014435
```

```
# Calculate SATE from God-mode
sate2 = mean(samp2$y1 - samp2$y0)
# Bias of difference in means
mean(dif.means)-sate2
```

```
## [1] 0.004172437
```

```
# Bias in linear regression
mean(lin.reg)-sate2
```

```
## [1] 0.00299457
```

```
# Efficiency of difference in means
var(dif.means)
```

```
## [1] 0.02683014
```

```r
# Efficiency of linear regression
var(lin.reg)
```

```
## [1] 0.02307917
```

With a resampled draw with a coefficient of 0.1 instead of 1.1, the bias of the estimators remain close to zero, still indicating that they are unbiased estimators. However, the efficiency of the difference in means method is now much closer, even indistinguishable than that of the linear regression. As the coefficient for the covariate decreases, the the difference in means estimator becomes more efficient.

Question 2

In a randomized block design, randomization occurs separately within blocks. In many situations, the ratio of treatment to control observations is different across blocks. In addition, the treatment effect may vary across sites. For this problem, you will simulate data sets for a randomized block design that includes a binary indicator for female as a blocking variable. You will then estimate the ATE with two estimators: one that accounts for the blocking structure and one that does not. You will compare the bias and efficiency of these estimators. We will walk you through this in steps.

(a) First simulate the blocking variable and potential outcomes for 100 observations. In particular:

- Set the seed to by 1234
- Generate female as blocking variable (Female vs. Other Ratio (30:70)
- Generate Y(0) and Y(1) with the following features: – the intercept is 70 – the residual standard deviation is 1.
  – treatment effect varies by block: observations with female=1 have treatment effect of 7 and those with female=0 have a treatment effect of 3. [Hint: Note that we are assuming that being female predicts treatment effect but does not predict the probability of being treated.]

```r
# Set seed
set.seed(1234)
# Generate female as blocking variable
female = rbinom(100,1,3/7)
# Generate y0 and y1
y0 = 70 + rnorm(100,0,1)
dftemp = data.frame(female=female,y0=y0)
dftemp = dftemp %>% mutate(y1 = if_else(female==1, 70 + 7, 70 + 3))
dftemp$y1 = dftemp$y1 + rnorm(100,0,1)
dat2 = dftemp
# Combine into a data frame
# Sanity check
dat2
```

```
##    female       y0       y1
## 1       0 68.19397 72.62276
## 2       1 69.41792 77.09762
## 3       1 68.89111 78.63874
## 4       1 68.98504 76.12441
## 5       1 69.83769 77.12176
## 6       1 70.56306 78.36213
## 7       0 71.64782 72.76538
## 8       0 69.22665 71.94662
## 9       1 71.60591 76.13022
## 10      0 68.84219 72.60987
## 11      1 70.65659 76.15265
```

```
## 12         0 72.54899 72.73936
## 13         0 69.96524 72.58558
## 14         1 69.33037 76.81695
## 15         0 69.99240 73.40706
## 16         1 71.77708 77.62463
## 17         0 68.86139 74.67821
## 18         0 71.36783 72.93131
## 19         0 71.32956 72.67916
## 20         0 70.33647 74.47101
## 21         0 70.00689 74.70433
## 22         0 69.54453 73.04324
## 23         0 69.63348 72.66734
## 24         0 70.64829 71.17776
## 25         0 72.07027 74.41126
## 26         1 69.84660 76.16242
## 27         0 68.60930 71.87624
## 28         1 69.27642 80.04377
## 29         1 70.25826 77.23502
## 30         0 69.68294 72.96674
## 31         0 69.82221 70.26778
## 32         0 69.83001 72.90021
## 33         0 68.62770 73.97603
## 34         0 69.82621 73.41387
## 35         0 70.85023 73.91232
## 36         1 70.69761 78.98373
## 37         0 70.55000 74.16911
## 38         0 69.59727 72.49126
## 39         1 69.80841 77.70418
## 40         1 68.80547 76.80158
## 41         0 69.94684 72.46193
## 42         1 70.25520 74.14424
## 43         0 71.70596 72.21035
## 44         1 71.00151 77.48781
## 45         0 69.50442 75.16803
## 46         0 70.35555 73.50069
## 47         1 68.86539 77.62021
## 48         0 70.87820 72.03410
## 49         0 70.97292 73.16265
## 50         1 72.12112 74.92176
## 51         0 70.41452 73.48523
## 52         0 69.52528 73.69677
## 53         1 70.06599 77.18551
## 54         0 69.49752 73.70073
## 55         0 69.17400 73.31168
## 56         0 70.16699 73.76046
## 57         0 69.10374 74.84246
## 58         1 70.16819 78.11236
## 59         0 70.35497 73.03266
## 60         1 69.94789 75.88555
## 61         1 69.80407 77.41806
## 62         0 69.35093 72.59976
## 63         0 68.89023 74.49349
## 64         0 70.84927 71.39292
## 65         0 70.02236 72.58425
```

```
## 66       1 70.83114 77.42201
## 67       0 68.75571 72.84826
## 68       0 70.16903 72.39385
## 69       0 70.67317 72.69528
## 70       0 69.97372 73.62954
## 71       0 69.80861 73.89517
## 72       1 69.21809 77.66021
## 73       0 72.05816 75.27348
## 74       1 70.75050 78.17350
## 75       0 71.82421 73.28771
## 76       0 70.08006 72.34023
## 77       0 69.36859 75.91914
## 78       0 68.48671 73.67742
## 79       0 69.36390 72.31568
## 80       1 70.22630 77.18649
## 81       1 71.01369 76.67561
## 82       0 70.25275 72.72530
## 83       0 68.82805 72.06650
## 84       0 70.66871 73.11685
## 85       0 68.34990 73.31916
## 86       1 69.63415 75.92246
## 87       0 69.68388 69.76685
## 88       0 68.05175 72.74513
## 89       0 70.92006 73.02952
## 90       1 69.37713 77.59427
## 91       0 69.66596 73.05914
## 92       1 71.39515 77.41340
## 93       0 70.63667 71.90223
## 94       0 69.89157 73.71118
## 95       0 70.51376 73.71889
## 96       0 70.39927 73.25165
## 97       0 71.66286 74.35727
## 98       0 70.27589 73.40447
## 99       0 70.50627 73.26436
## 100      1 70.34755 77.26804
```

(b) Calculate the overall SATE and the SATE for each block

```
# Calculate overall SATE
mean(dat2$y1-dat2$y0)
```

```
## [1] 4.336822
```

```
# Calculate female SATE
mean(dat2$y1[dat2$female==1]-dat2$y0[dat2$female==1])
```

```
## [1] 7.00971
```

```
# Calculate male SATE
mean(dat2$y1[dat2$female==0]-dat2$y0[dat2$female==0])
```

```
## [1] 3.078992
```

Now create a function for assigning the treatment In particular: * Within each block create different assignment probabilities:

$$\Pr(Z = 1 \mid \text{female} = 0) = .6 \Pr(Z = 1 \mid \text{female} = 1) = .4$$

```r
# Start function
dgp2.fun = function(){
  df = dat2
  # Import female column from data
  # Assign treatment
  df[df$female == 1, "treatment"] = rbinom(nrow(df[df$female==1,]), 1, 0.4)
  df[df$female == 0, "treatment"] = rbinom(nrow(df[df$female==0,]), 1, 0.6)
  # Combine into data frame
  # Use mutate from tidyverse to create y's
  df = df %>% mutate(y = if_else(df$treatment==1, y1, y0))
  result = df %>% select(-c(y1,y0))

  return(result)
}
```

Generate the treatment and create a vector for the observed outcomes implied by that treatment.

```r
# Draw data
with.treat = dgp2.fun()
# Quick sanity check
with.treat %>% group_by(female) %>% count(treatment)
```

```
## # A tibble: 4 x 3
## # Groups:   female [2]
##   female treatment     n
##    <int>     <int> <int>
## 1      0         0    37
## 2      0         1    31
## 3      1         0    14
## 4      1         1    18
```

```r
with.treat
```

```
##     female treatment        y
## 1        0         1 72.62276
## 2        1         1 77.09762
## 3        1         1 78.63874
## 4        1         1 76.12441
## 5        1         0 69.83769
## 6        1         1 78.36213
## 7        0         0 71.64782
## 8        0         1 71.94662
## 9        1         1 76.13022
## 10       0         0 68.84219
## 11       1         1 76.15265
## 12       0         0 72.54899
## 13       0         0 69.96524
## 14       1         1 76.81695
## 15       0         0 69.99240
## 16       1         0 71.77708
## 17       0         1 74.67821
## 18       0         1 72.93131
## 19       0         0 71.32956
## 20       0         0 70.33647
## 21       0         1 74.70433
## 22       0         0 69.54453
```

```
## 23        0          1 72.66734
## 24        0          0 70.64829
## 25        0          0 72.07027
## 26        1          1 76.16242
## 27        0          0 68.60930
## 28        1          0 69.27642
## 29        1          1 77.23502
## 30        0          0 69.68294
## 31        0          1 70.26778
## 32        0          1 72.90021
## 33        0          0 68.62770
## 34        0          1 73.41387
## 35        0          0 70.85023
## 36        1          1 78.98373
## 37        0          0 70.55000
## 38        0          1 72.49126
## 39        1          0 69.80841
## 40        1          0 68.80547
## 41        0          1 72.46193
## 42        1          1 74.14424
## 43        0          0 71.70596
## 44        1          1 77.48781
## 45        0          0 69.50442
## 46        0          1 73.50069
## 47        1          0 68.86539
## 48        0          0 70.87820
## 49        0          0 70.97292
## 50        1          1 74.92176
## 51        0          0 70.41452
## 52        0          1 73.69677
## 53        1          0 70.06599
## 54        0          0 69.49752
## 55        0          0 69.17400
## 56        0          0 70.16699
## 57        0          1 74.84246
## 58        1          1 78.11236
## 59        0          1 73.03266
## 60        1          0 69.94789
## 61        1          0 69.80407
## 62        0          0 69.35093
## 63        0          0 68.89023
## 64        0          1 71.39292
## 65        0          0 70.02236
## 66        1          1 77.42201
## 67        0          1 72.84826
## 68        0          1 72.39385
## 69        0          1 72.69528
## 70        0          1 73.62954
## 71        0          1 73.89517
## 72        1          0 69.21809
## 73        0          0 72.05816
## 74        1          1 78.17350
## 75        0          0 71.82421
## 76        0          1 72.34023
```

```
## 77       0         1 75.91914
## 78       0         1 73.67742
## 79       0         1 72.31568
## 80       1         0 70.22630
## 81       1         0 71.01369
## 82       0         1 72.72530
## 83       0         1 72.06650
## 84       0         0 70.66871
## 85       0         0 68.34990
## 86       1         1 75.92246
## 87       0         0 69.68388
## 88       0         0 68.05175
## 89       0         0 70.92006
## 90       1         0 69.37713
## 91       0         1 73.05914
## 92       1         0 71.39515
## 93       0         1 71.90223
## 94       0         1 73.71118
## 95       0         0 70.51376
## 96       0         0 70.39927
## 97       0         0 71.66286
## 98       0         0 70.27589
## 99       0         1 73.26436
## 100      1         1 77.26804
```

```
# Create a single vector with all outcomes for treatment == 1
obs.outcome = with.treat$y[with.treat$treatment==1]
obs.outcome
```

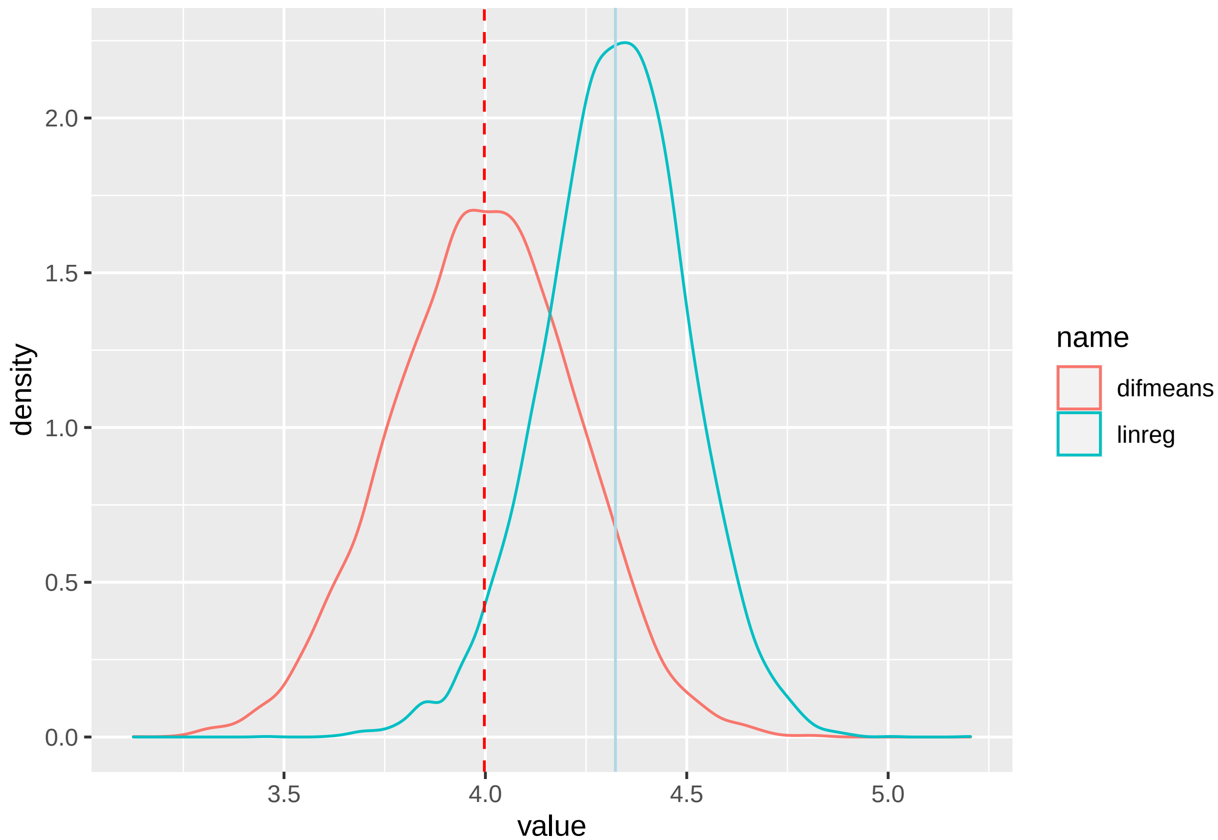
```
##  [1] 72.62276 77.09762 78.63874 76.12441 78.36213 71.94662 76.13022 76.15265
##  [9] 76.81695 74.67821 72.93131 74.70433 72.66734 76.16242 77.23502 70.26778
## [17] 72.90021 73.41387 78.98373 72.49126 72.46193 74.14424 77.48781 73.50069
## [25] 74.92176 73.69677 74.84246 78.11236 73.03266 71.39292 77.42201 72.84826
## [33] 72.39385 72.69528 73.62954 73.89517 78.17350 72.34023 75.91914 73.67742
## [41] 72.31568 72.72530 72.06650 75.92246 73.05914 71.90223 73.71118 73.26436
## [49] 77.26804
```

We will use this to create a randomization distribution for two different estimators for the SATE. Obtain 10,000 draws from that distribution.

```
# Define iterations
ITER = 10000
# Initalize results vectors
difmeansresults = rep(NA,ITER)
linregresults = rep(NA,ITER)
# Begin loop for 10000 draws
for (i in 1:ITER) {
  loopdata = dgp2.fun()
  difmeansresults[i] = mean(loopdata$y[loopdata$treatment==1])-mean(loopdata$y[loopdata$treatment==0])
  linregresults[i] = summary(lm(y~female+treatment,loopdata))$coefficients[3]
}
```

(c) Plot the (Monte Carlo estimate of the) randomization distribution for each of the two estimators: difference in means and regression. (Note: Similar to Problem 1, the difference in means estimator will ignore blocks and the regression estimator will adjust for the blocks.) Either overlay the two plots (with different colors for each) or make sure the xlim on both plots is the same.

```r
# Combine our results into a data frame
df2 = data.frame(difmeans=difmeansresults,linreg=linregresults)
# Using tidyverse and ggplot to graph the difference in means and linear regression draws
# Pivot the dataframe to stack all values in one column
df2 %>% pivot_longer(1:2) %>%
  # Call ggplot and color by name
  ggplot(aes(value,color=name)) +
    # Graph density plots of both dif.means and lin.reg
    geom_density() +
    geom_vline(xintercept = mean(df2$difmeans),color="red",linetype="dashed") +
    geom_vline(xintercept = mean(df2$linreg),color="light blue")
```



(d) Calculate the bias and efficiency of each estimator. Also calculate the root mean squared error.

```r
# Calculate SATE from God-mode
sate2 = mean(dat2$y1 - dat2$y0)
# Bias of difference in means
mean(difmeansresults)-sate2
```

```
## [1] -0.3393254
```

```r
# Bias in linear regression
mean(linregresults)-sate2
```

```
## [1] -0.01399301
```

```r
# Efficiency of difference in means
var(difmeansresults)
```

```
## [1] 0.05228862
```

```r
# Efficiency of linear regression
var(linregresults)
```

```
## [1] 0.03188399
```

```r
# Root mean squared error difference in means
sqrt(mean(sate2-difmeansresults)^2)
```

```
## [1] 0.3393254
```

```r
# Root mean squared error linear regression
sqrt(mean(sate2-linregresults)^2)
```

```
## [1] 0.01399301
```

(e) Why is the estimator that ignores blocks biased? Is the efficiency meaningful here? Why did I have you calculate the RMSE? An estimator that ignores blocks is biased if the outcomes are conditioned on meaningful blocks. The effect of the treatment is conditional on female distribution and female distribution is not even across the sample.

(f) Describe one possible real-life scenario where treatment assignment probabilities and/or treatment effects vary across levels of a covariate. In public education, assignment probabilities for, let's say, supplementary English language arts programs, the treatment effects will vary greatly on ELL and LEP students as opposed to those already profiecient in English. In this situation, an analyst would have to take care to separate these blocks, else inaccurately estimate treatment effects.

(g) How could you use a regression to estimate the treatment effects separately by group? Calculate estimates for our original sample and treatment assignment (with seed 1234). To see separate treatment effects by group, we can introduce interaction terms to the regression.

```r
# Let's pull another set of data
with.int = dgp2.fun()
reg1 = lm(y~female*treatment,with.int)
# Estimate for female, treated
summary(reg1)$coefficient[1]+summary(reg1)$coefficient[2]+summary(reg1)$coefficient[3]+summary(reg1)$co
```

```
## [1] 77.42629
```

```r
# Estimate for male, treated
summary(reg1)$coefficient[1]+summary(reg1)$coefficient[3]
```

```
## [1] 73.06564
```

```r
# Estimate for female, untreated
summary(reg1)$coefficient[1]+summary(reg1)$coefficient[2]
```

```
## [1] 70.12856
```

```r
# Estimate for male, untreated
summary(reg1)$coefficient[1]
```

```
## [1] 69.8788
```