CS112 Assignment 3, Fall 2020

Enter your name here

11/01/2020

A few important notes

Option 1 for submitting your assignment: *This method is actually preferred. This is an RMarkdown document. Did you know you can open this document in RStudio, edit it by adding your answers and code, and then knit it to a pdf? To submit your answers to this assignment, simply knit this file as a pdf and submit it as a pdf on Forum. All of your code must be included in the resulting pdf file, i.e., don't set echo = FALSE in any of your code chunks. To learn more about RMarkdown, watch the videos from session 1 and session 2 of the CS112B optional class. This is also a cheat sheet for using Rmarkdown. If you have questions about RMarkdown, please post them on Perusall. Try knitting this document in your RStudio. You should be able to get a pdf file. At any step, you can try knitting the document and recreate a pdf. If you get error, you might have incomplete code.*

Option 2 for submitting your assignment: *If you are not comfortable with RMarkdown, you can also choose the Google Doc version of this assignment, make a copy of it and edit the Google doc (include your code, figures, results, and explanations) and at the end download your Google Doc as a pdf and submit the pdf file.*

Note: *Either way (if you use Rmd and knit as pdf OR if you use Google Doc and download as pdf) you should make sure you put your name on top of the document.*

Note: *The first time you run this document you may get error that some packages don't exist. If you don't have the packages listed on top of this document, install them first and you won't get those errors.*

Note: *If you work with others or get help from others on the assignment, please provide the names of your partners at the top of this document. Working together is fine, but all must do and understand their own work and have to mention the collaboration on top of this document.*

Note: *Don't change seed in the document. The function `set.seed()` has already been set at the beginning of this document to 928. Changing the see again to a different number will make your results not replicable.*

QUESTION 1: Data Generating Example

The following code, creates a toy dataset with a treatment variable, D, an outcome variable, Y, and other variables V_1 to V_4.

```r
n = 1000
## Generating a random data set here
#Syntax for the normal distribution here is rnorm(sample size, mean, SD)
V1 = rnorm(n, 45, 10)
#getting a binary variable
V2 = sample(c(1,0),
             replace = TRUE,
             size = n,
             prob = c(.4,.6))
V3 = rnorm(n, V1/10, 1)
```

```r
V4 = rnorm(n, 0, 1)
D  = as.numeric(pnorm(rnorm(n, .01*V1 + .8*V2 + 0.3*V3 + V4, 1), .45 + .32 + .3*4.5, 1) > .5)
Y  = rnorm(n, .8*D - 0.45*V2 - .4*V3 + 2, .2)
# combining everything in a data frame
df = data.frame(V1, V2, V3, V4, D, Y)
```

STEP 1 From the variables V_1, V_2, V_3, and V_4, which one(s) are not confounding variable(s) (covariates that causes confounding)? Remember, a rule of thumb (although not a perfect rule) is that the variable is correlated with both the treatment variable and the outcome variable. Explain!

Confounding variables are those that are correlated with both the Treatment and Outcome variable. A simple example is if we associate sales of ice-cream with number of shark attacks, we will find a spurious correlation between the events. However, these two events are correlated because of the confounding variable - summer (or increase of temperature). As the summer comes, more people come the the beaches and buy ice-cream there. However, more people at the beaches means a higher proportion of people that would be attacked by sharks.

Now, let's turn back to our dataframe and identify confounding variables. We will consider a correlation over 0.3 a moderate one: 1. V3 is moderately correlated with treatment and strongly negatively correlated with the Outcome. 2. V4 is moderately correlated with Treatment and Outcome variables 3. The rest of the variables seem to have a weak correlation Therefore, we consider V3 and V4 to be strong confounding variables.

```r
cor(df)
```

```
##             V1           V2          V3           V4         D          Y
## V1  1.00000000  0.078094128  0.70029682 -0.014311665 0.2312331 -0.5162818
## V2  0.07809413  1.000000000  0.01598732  0.009587117 0.2450834 -0.2077495
## V3  0.70029682  0.015987317  1.00000000  0.011386732 0.2919576 -0.7278276
## V4 -0.01431166  0.009587117  0.01138673  1.000000000 0.5394971  0.3301658
## D   0.23123308  0.245083401  0.29195764  0.539497059 1.0000000  0.2715761
## Y  -0.51628176 -0.207749511 -0.72782756  0.330165793 0.2715761  1.0000000
```

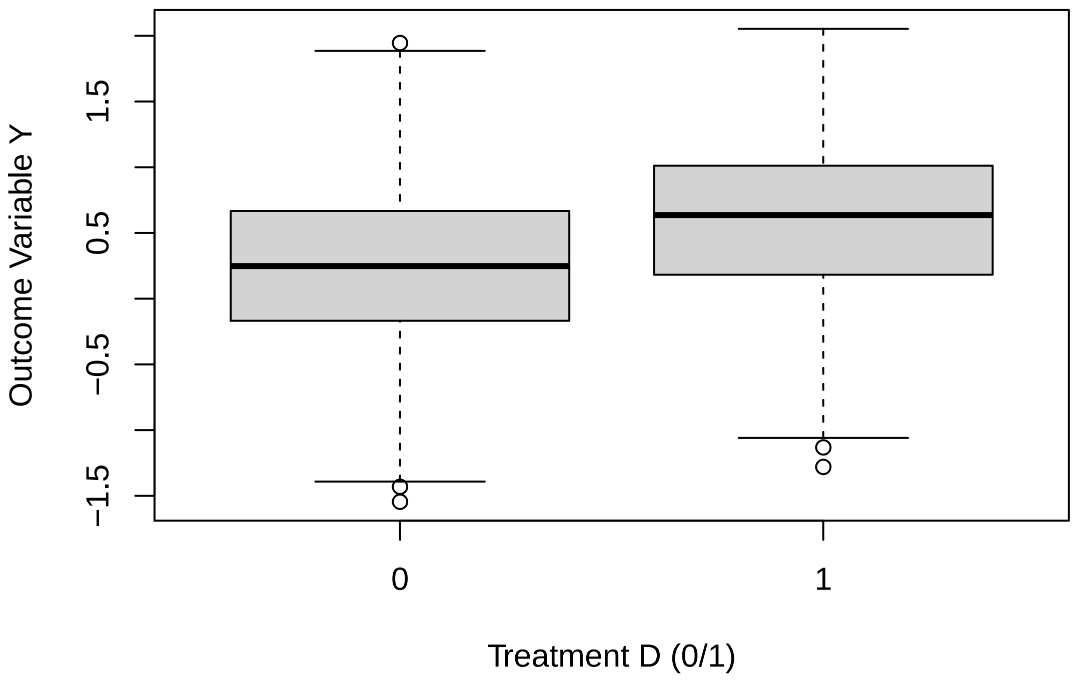
STEP 2 Can you figure out the true treatment effect by looking at the data generating process above?

The true treatment effect is 0.8 as can be seen in line generating the outcome variable Y

STEP 3 Plot the outcome variable against the treatment variable. Make sure you label your axes. Do you see a trend?

We can see a trend that units who received the treatment have a higher outcome, on average.

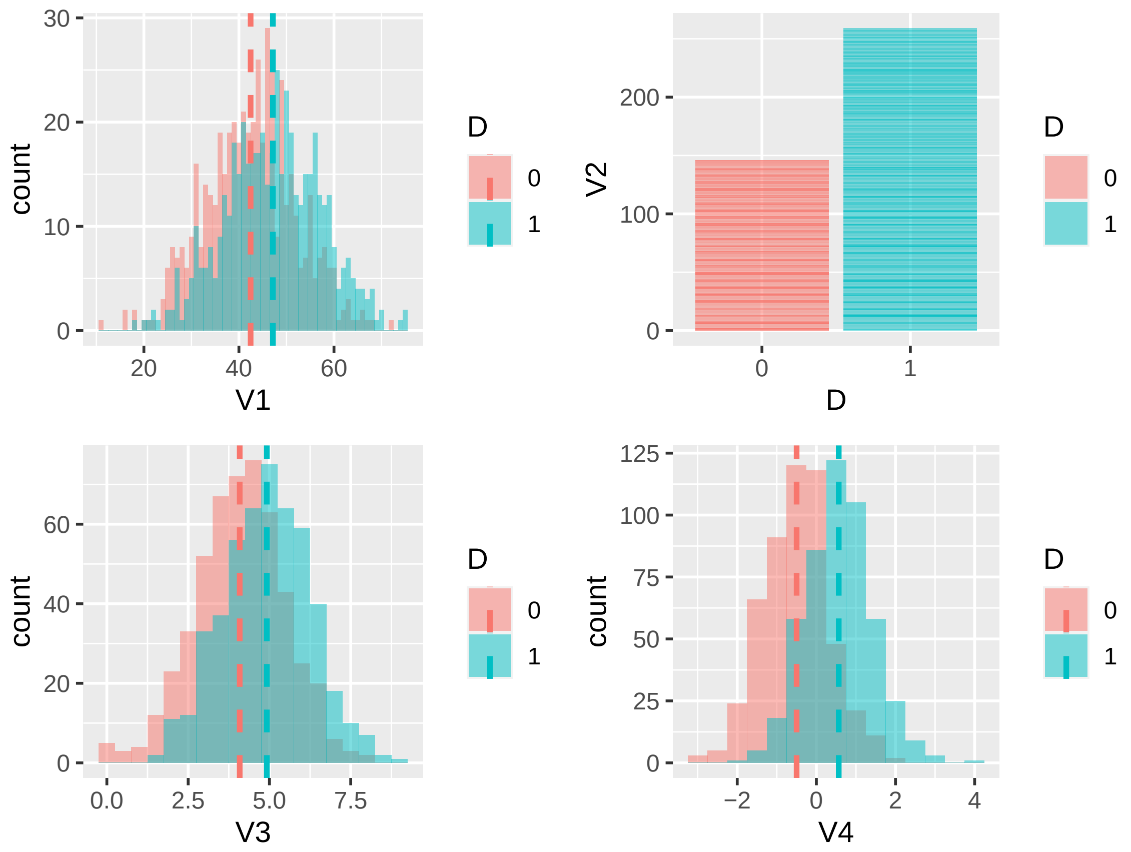
```r
# Your code here

plot(as.factor(df$D), df$Y, xlab="Treatment D (0/1)", ylab="Outcome Variable Y")
```



STEP 4 Are the variables V_1, V_2, V_3, and V_4 balanced across the treatment and control groups? You can use any R function from any package to check this (for instance, you can check the cobalt package). Make sure you check all variables.

Note: *This is optional but you can use the `gridExtra` package and its `grid.arrange()` function to put all the 4 graphs in one 2 x 2 graph. Read more about the package and how to use it here: https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html. Set `nrow = 2`.*

```
# Your code here

df2 <- df %>%
  mutate(D = as.factor(D))

dfstat <- ddply(df2, "D", summarise, V1.mean=mean(V1), V2.mean=mean(V2), V3.mean=mean(V3), V4.mean=mean

V1_plot <- ggplot(df2, aes(x=V1, fill=D)) +
    geom_histogram(binwidth=1, alpha=.5, position="identity") +
    geom_vline(data=dfstat, aes(xintercept=V1.mean,  colour=D),
               linetype="dashed", size=1)

V2_plot <- ggplot(df2, aes(x=D, y=V2, fill=D)) +
    geom_col(alpha=.5)

V3_plot <- ggplot(df2, aes(x=V3, fill=D)) +
    geom_histogram(binwidth=.5, alpha=.5, position="identity") +
    geom_vline(data=dfstat, aes(xintercept=V3.mean,  colour=D),
                linetype="dashed", size=1)

V4_plot <- ggplot(df2, aes(x=V4, fill=D)) +
    geom_histogram(binwidth=.5, alpha=.5, position="identity") +
    geom_vline(data=dfstat, aes(xintercept=V4.mean,  colour=D),
                linetype="dashed", size=1)


grid.arrange(V1_plot, V2_plot, V3_plot, V4_plot, nrow = 2)
```



STEP 5 Write code that would simply calculate the Prima Facie treatment effect in the data above. What's the Prima Facie treatment effect? Note that the outcome variable is binary.

```
# Your code here
tr <- df %>% filter(D == 1)
ctr <- df %>% filter(D == 0)

#mean(df$Y[df$D == 1]) - mean(df$Y[df$D == 0])

mean(tr$Y) - mean(ctr$Y)
```

```
## [1] 0.3423726
```

STEP 6 Explain why the Prima Facie effect is not the true average causal effect from Step 2.

From Step 5, we can see that the Prima Facie effect (the estimated treatment effect) is 0.42, while the true treatment effect is 0.8 (from step 2). This happens because of the self-selection bias, which is essentially the basic inherent difference between the treatment and control groups. As we saw in the balance plots above, the mean values for all variables are larger for the treatment group.

For example, mean of V1 for treatment is 47, for control it is 42. From the correlation table, we see that V1 is negatively correlated (-0.5) with the outcome variable. Therefore, while treatment D increases Y, the increase in V1 **decreases** Y. This confuses the results and we cannot attribute the difference in Y means to just the treatment D.

One solution to that is using Matching, which pairs treatment to control units that are similar or identical. If the 2 units are absolutely identical with the only difference in D, then we can attribute the change in Y to D, giving us a more accurate treatment effect. We still do not call it the true treatment effect because even thought the units are identical, the environment around them might be changing, such as where the treatment was given, how their bodies reacted, what time of the day the treatment was give. All these concerns might result in different treatment effect for two units. Therefore, the only way to solve the Fundamental Problem of Causal Problem is if we had 2 parallel universes.

STEP 7 We can use matching to create a better balance of the covariates. Use a propensity score model that includes all the variables V_1, V_2, V_3, and V_4.

```
# Your code here
lm1 <- glm(D ~ V1 + V2 + V3 + V4, data=df, family=binomial)

df$pscore <- predict(lm1, type="response")

df_match <- Match(Y=df$Y, Tr=df$D, X=df$pscore, M=1)

#mb2 <- MatchBalance(D ~ V1 + V2 + V3 + V4, data=df, match.out=df_match, nboots=500)
```

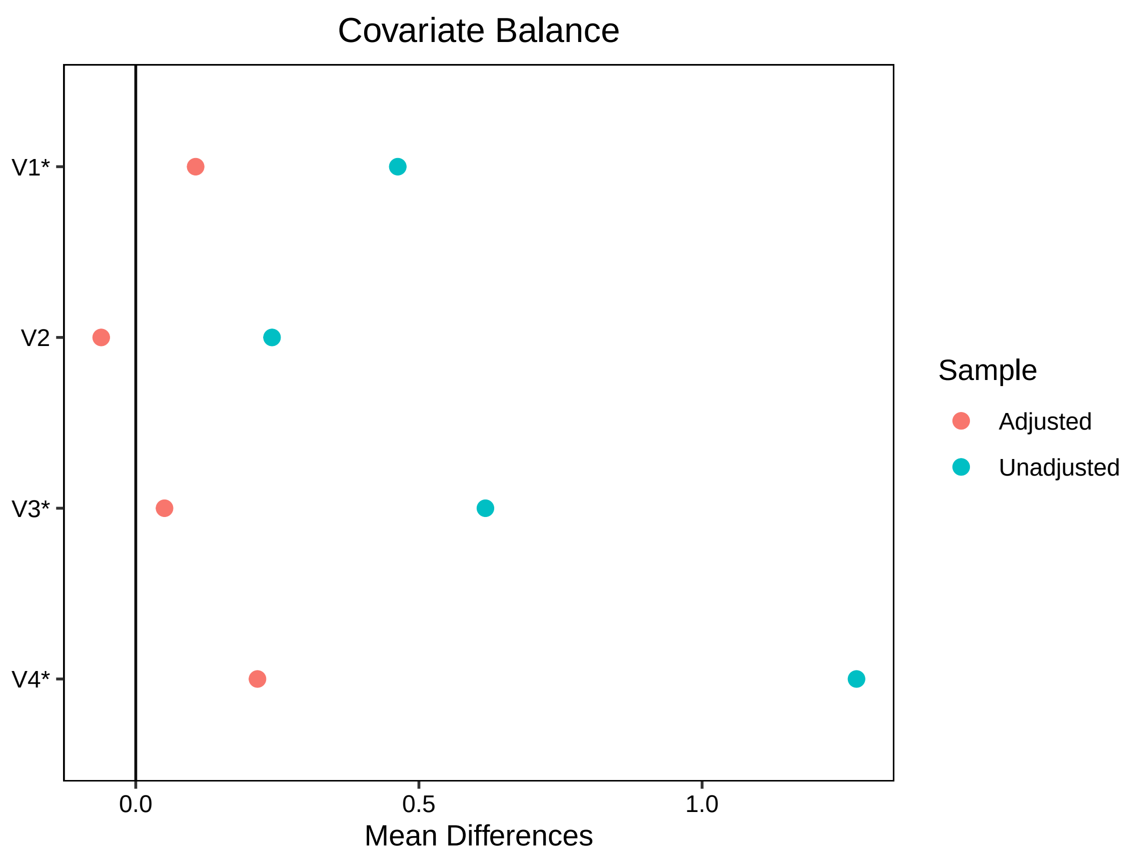
STEP 8 Check the balance of the covariates. Do you see any improvements after matching?

As we can see from the output of bal.tab and the love plot, the difference in treatment and control groups for each covariate has decreases. However, looking at just the means is not sufficient because it does not tell us anything about the distribution. Hence, we can also look at bal.plot for variable V4: we can see a significant improvement after matching because the distributions look more similar and are centered at almost the same number.

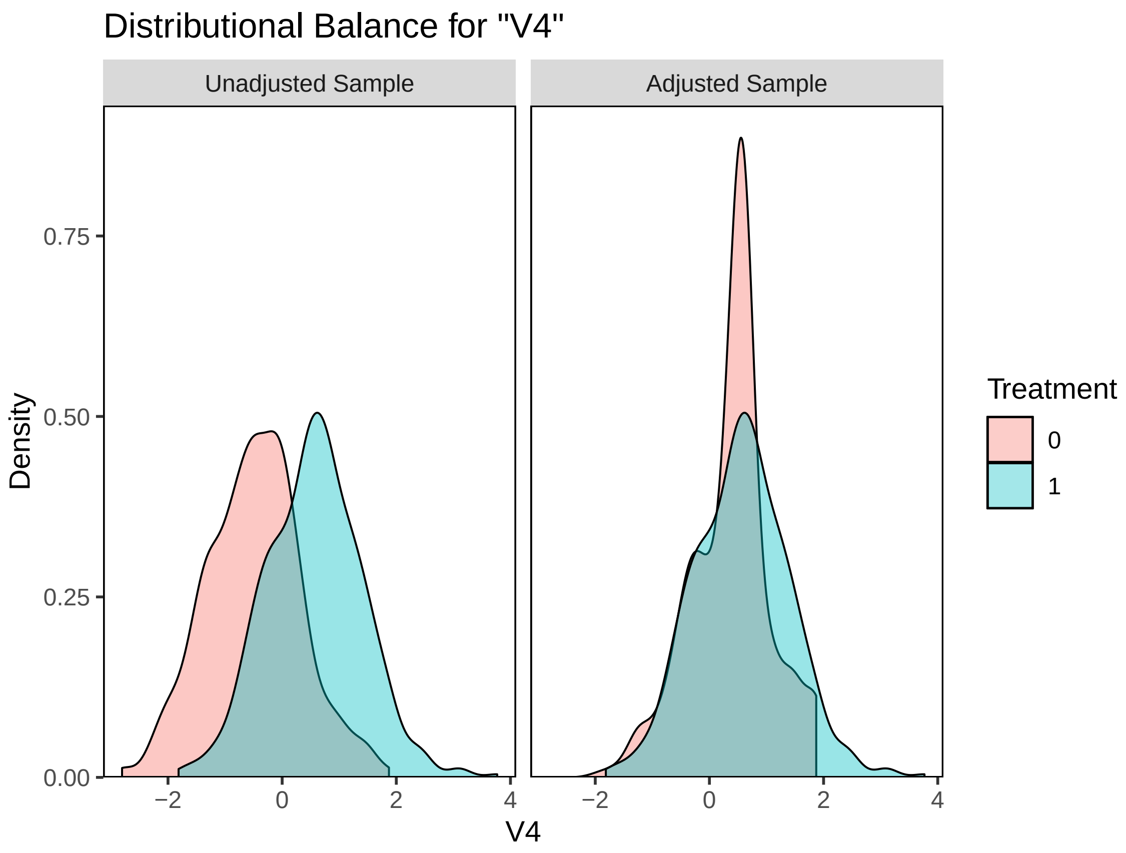
```
# Your code here
bal.tab(df_match, formula = (D ~ V1 + V2 + V3 + V4), data = df, distance = ~ lm1$fitted.values)
```

```
## Balance Measures
##                        Type Diff.Adj
## lm1$fitted.values Distance   0.0358
## V1                  Contin.   0.1057
## V2                   Binary  -0.0610
## V3                  Contin.   0.0508
## V4                  Contin.   0.2149
##
## Sample sizes
##                       Control Treated
## All                     509.      491
## Matched (ESS)            10.84    491
## Matched (Unweighted)    224.      491
## Unmatched               285.        0
```

```
love.plot(x=df_match, formula = (D ~ V1 + V2 + V3 + V4), data = df,method = "weighting", estimand = "AT
```



```r
bal.plot(x=df_match, var.name = "V4", which="both", formula = (D ~ V1 + V2 + V3 + V4))
```



STEP 9 What is the treatment effect after matching? Is this surprising given your answer to Step 2. Is the treatment effect found in this Step closer to the treatment effect in Step 2 than the treatment effect before matching?

Now, the treatment effect is much closer to the true treatment effect. We can see it by looking at the 95% CI, which almost includes the true treatment effect of 0.8.

```
# Your code here
df_match$se[1]
```

```
## [1] 0.1622071
```

```
lwr <- df_match$est[1] - df_match$se[1]
upr <- df_match$est[1] + df_match$se[1]

sprintf("Treatment effect: %f, 95%% Confidence Interval: [%f, %f]", df_match$est[1], lwr, upr)
```

```
## [1] "Treatment effect: 0.813256, 95% Confidence Interval: [0.651049, 0.975463]"
```

QUESTION 2: Daughters

Read Section 5 (which is only about 1 page long!) of Iacus, King, Porro (2011), Multivariate Matching Methods That Are Monotonic Imbalance Bounding, JASA, V 106, N. 493, available here: https://gking.harvard.edu/files/gking/files/cem_jasa.pdf. Don't worry about the "CEM" elements. Focus on the "daughters" case study.

Data for this case study is available in "doughters" below.

```r
daughters = read.csv(url("http://bit.ly/daughters_data")) %>%
  clean_names()
```

STEP 1 Before doing any matching, run a regression, with the outcome variable being `nowtot`, the treatment variable being `hasgirls`, and the independent vars mentioned below: - dems, - repubs, - christian, - age, - srvlng, - demvote

Show the regression specification. Use the regression to estimate a treatment effect and confidence interval. Check the balance of this not-matched data set using any method of your choice (balance tables, balance plots, love plots, etc).

We will look at the balance of variables after we match in the next steps.

```r
# Your code here,
lm2 <- lm(nowtot ~ hasgirls + dems + repubs + christian + age + srvlng + demvote, data=daughters)

sed <- summary(lm2)$coefficients[, 2][2]
coefd <- lm2$coefficients[2][1]
lwrb <- coefd - sed
uprb <- coefd + sed

# Treatment effect - coefficient for the treatment variable hasgirls
sprintf("The Treatment Effect for hasgirls is %f, with 95%% CI [%f, %f]", coefd, lwrb, uprb)
```

```
## [1] "The Treatment Effect for hasgirls is -0.452268, with 95% CI [-2.355905, 1.451370]"
```

STEP 2 Then, do genetic matching. Use the same variables as in the regression above. Make sure to set `estimand = "ATT"`. What's your treatment effect?

Note: *For replicability purposes, we need to choose a seed for the* `GenMatch()` *function. However, setting seed for* `GenMatch()` *is different. The usual practice of typing* `set.seed(some number)` *before the GenMatch line doesn't ensure stochastic stability. To set seeds for* `GenMatch`*, you have to run* `GenMatch()` *including instructions for genoud within* `GenMatch()`*, e.g.:* `GenMatch(Tr, X, unif.seed = 123, int.seed = 92485...)`*. You can find info on these* `.seed elements` *in the documentation for* `genoud()`*. The special .seed elements should only be present in the* `GenMatch()` *line, not elsewhere (because nothing besides* `GenMatch()` *runs genoud.*

Note: *When you use the* `GenMatch()` *function, wrap everything inside the following function* `invisible(capture.output())`*. This will reduce the unnecessary output produced from the GenMatch() function. For instance, you can say:* `invisible(capture.output(genout_daughters <- GenMatch(...)))`

```r
# Your code here
library(rgenoud)
Y <- daughters$nowtot
Tr <- daughters$hasgirls
X <- cbind(daughters$dems,daughters$repubs, daughters$christian, daughters$age, daughters$srvlng, daugh

invisible(capture.output(genout2 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=1, pop.size = 30, max.gene

m.out2 <- Match(Y = Y, Tr=Tr, X=X, M=1, Weight.matrix = genout2)

m.out2$est
```

```
##           [,1]
## [1,] 0.7291667
```

STEP 3 Summarize (in 5-15 sentences) the genetic matching procedure and results, including what you matched on, what you balanced on, and what your balance results were. Provide output for MatchBalance() in the body of your submission.

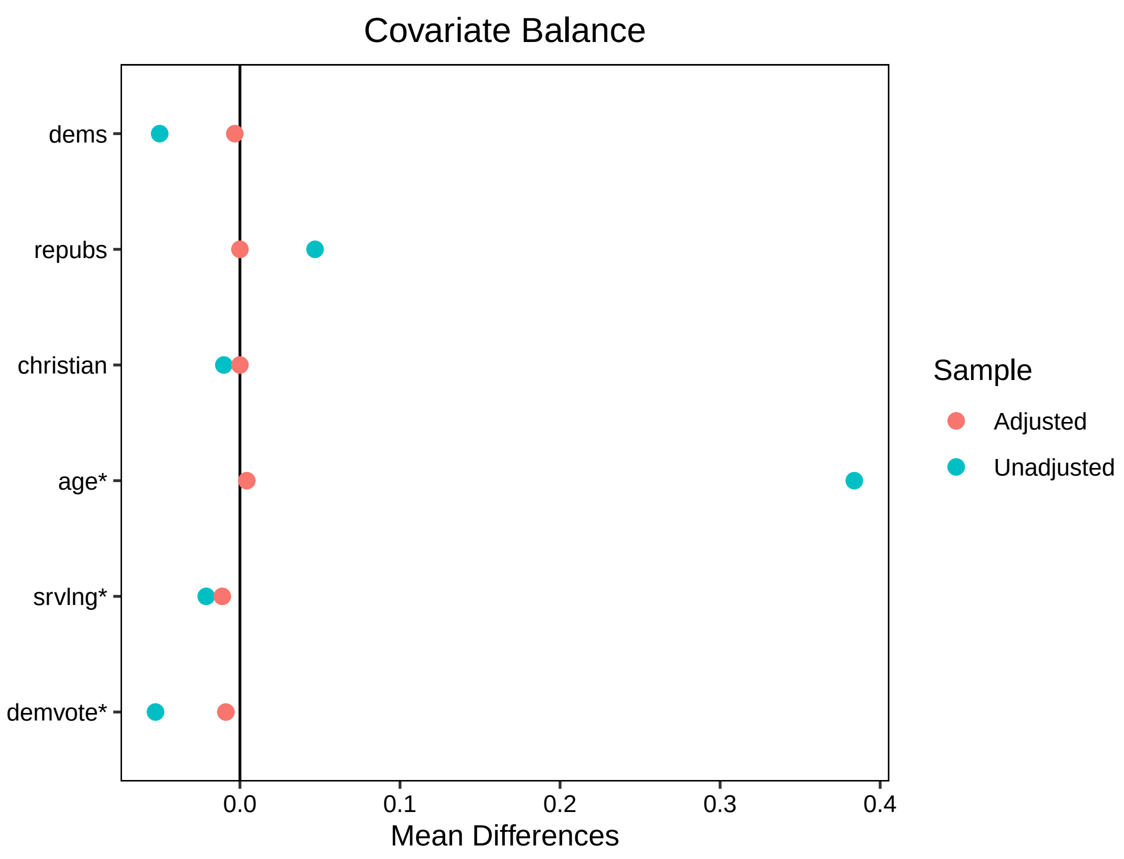
When we use distance (e.g. Euclidian) to match units, the scale we pick for each variable matters. The quality of matching depends on the scale we choose, so it is in out interest to find the optimal scale for each variable. Genetic Matching Algorithm is a smart-search algorithm that looks for the optimal set of scales: 1. It starts with a random set of scales. 2. Scales that result in a good balance of variables proceed to the next generation. Balance can be measured with statistics such as ks-statistics which quantitatively represents the quality balance between distributions of treatment and control. 3. Some scales are randomly modified to increase variability in species. 4. After the specificied number of generations, scale that results in the optimal balance is chosen.

In our case, we matched and balanced on variables dems, repubs, christian, age, srvlng, and demvote. By looking at the output of bal.tab and love.plot, we can see that genetic matching resulted in a good balance of covariates. As we mentioned before, it is also useful to look at the distributions of the variables rather than just the means. Hence, we also examine at least one distribution using bal.plot below.

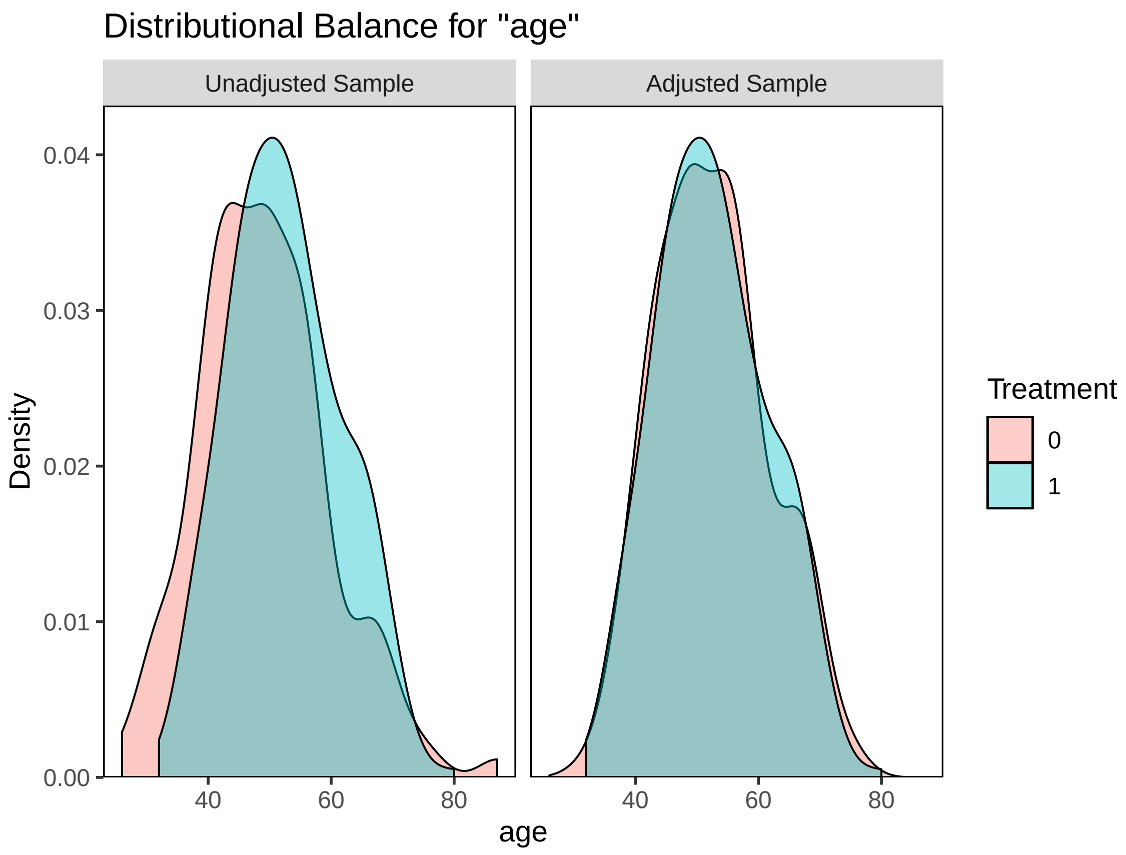
```
bal.tab(m.out2, formula = (hasgirls ~ dems + repubs + christian + age + srvlng + demvote), data = daugh
```

```
## Balance Measures
##              Type Diff.Adj
## dems       Binary  -0.0032
## repubs     Binary   0.0000
## christian  Binary   0.0000
## age       Contin.   0.0043
## srvlng    Contin.  -0.0110
## demvote   Contin.  -0.0088
##
## Sample sizes
##                      Control Treated
## All                    118.      312
## Matched (ESS)           66.7     312
## Matched (Unweighted)    93.      312
## Unmatched               25.        0
```

```
love.plot(x=m.out2, formula = (hasgirls ~ dems + repubs + christian + age + srvlng + demvote), data = da
```



```
bal.plot(x=m.out2, data=daughters, var.name = "age", which="both", formula = (hasgirls ~ dems + repubs
```



STEP 4 Is your treatment effect different from the one reported before matching? By how much? If your numbers are different, provide some explanation as to why the two numbers are different. If they're not, provide an explanation why they're not different.

The treatment effect after matching is 0.78, which is different from the one we obtained through linear regression before matching (-0.45). For each girl, the politician's total Women Support Score increases by 0.78 points. The results are different because after matching, we obtained a better balance of covariates between the treatment and control groups. This decreased the self-selection bias, making our causal inferences less biased and more accurate.

```r
coefd2 <- m.out2$est
sed2 <- m.out2$est
lwrd2 <- coefd2 - sed2
uprd2 <- coefd2 + sed2
sprintf("The Treatment Effect for hasgirls is %f, 95%% CI [%f, %f]", coefd2, lwrd2, uprd2)
```

```
## [1] "The Treatment Effect for hasgirls is 0.729167, 95% CI [0.000000, 1.458333]"
```

STEP 5 Change the parameters in your genetic matching algorithm to improve the balance of the covariates. Consider rerunning with M = 2 or 3 or more. Consider increasing other parameters in the `GenMatch()` function such as the number of generations and population size, caliper, etc. Try 10 different ways but don't report the results of the genetic matching weights or the balance in this document. Only report the treatment effect of your top 3 matches. For instance, run the `Match()` function three times for your top 3 genout objects. Make sure the summary reports the treatment effect estimate, the standard error, and the confidence interval. Do you see a large variation in your treatment effect between your top 3 models?

Note: *For replicability purposes, we need to choose a see for the `GenMatch()` function. However, setting seed* *for `GenMatch()` is different. The usual practice of typing `set.seed(123)` before the GenMatch line doesn't ensure stochastic stability. To set seeds for `GenMatch`, you have to run `GenMatch()` including instructions for genoud within `GenMatch()`, e.g.: `GenMatch(Tr, X, unif.seed = 123, int.seed = 92485...)`. You can find info on these `.seed elements` in the documentation for `genoud()`. The special .seed elements should only be present in the `GenMatch()` line, not elsewhere (because nothing besides `GenMatch()` runs genoud.*

Note: *When you use the `GenMatch()` function, wrap everything inside the following function `invisible(capture.output())`. This will reduce the unnecessary output produced with the GenMatch() function. For instance, you can say: `invisible(capture.output(genout_daughters <- GenMatch(...)))`*

Note: *In the matching assignment, you may find that the Genetic Matching step takes a while, e.g., hours. If you have to reduce pop.size to e.g., 10 or 16 to ensure it stops after only an hour or two, that's fine. Running your computer for an hour or two is a good thing. Running it for a full day or more is unnecessary overkill (and if this is your situation, change hyperparameters like pop.size to reduce run-time). For example, we suggest you modify the pop.size (e.g., you can set it to 5!), max.generations (set it to 2!), and wait.generations (set it to 1!) and that should expedite things.*

Note: *Can you set a caliper for one confounding variable, and not others (e.g., only set a caliper for "age")? No and yes. No, strictly speaking you can't. But yes, practically speaking you can, if you set other calipers (for the other confounders) that are so wide as to not induce any constraints. E.g., in GenMatch, and Match, you could set `caliper = c(1e16, 1e16, 0.5, 1e16)` and this would induce a certain meaningful caliper for the third confounder in X, without constraining the other confounders (because 1e16 implies a caliper that is so enormously wide that it does not, in practical terms, serve as a caliper at all).*

```
# Your code here
# -- 1. -- M=3
invisible(capture.output(genout3 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=3, pop.size = 16, max.gene

m.out3 <- Match(Y = Y, Tr=Tr, X=X, M=3, Weight.matrix = genout3)

# -- 2. -- M=3, caliper =
invisible(capture.output(genout4 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", replace=TRUE, M=3, pop.size =

m.out4 <- Match(Y = Y, Tr=Tr, X=X, M=3, Weight.matrix = genout4, replace=TRUE, caliper = c(1e16, 1e16, 

# -- 3. -- M=2, caliper = contrained
invisible(capture.output(genout5 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", replace=TRUE, M=3, pop.size =

m.out5 <- Match(Y = Y, Tr=Tr, X=X, M=3, Weight.matrix = genout5, replace=TRUE, caliper =  c(0.5, 0.5, 0

# -- 4. --
invisible(capture.output(genout6 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=2, pop.size = 32, max.gene

m.out6 <- Match(Y = Y, Tr=Tr, X=X, M=2, Weight.matrix = genout6)

# -- 5. --
invisible(capture.output(genout7 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=2, pop.size = 16, max.gene

m.out7 <- Match(Y = Y, Tr=Tr, X=X, M=2, Weight.matrix = genout7, caliper = c(0.5, 1e16, 0.5, 0.5,0.5,1e

# -- 6. --
invisible(capture.output(genout8 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=2, pop.size = 16, max.gene

m.out8 <- Match(Y = Y, Tr=Tr, X=X, M=2, Weight.matrix = genout8, caliper = c(0.5, 1e16, 1e16, 1e16,1e16
```

```r
# -- 7. --
invisible(capture.output(genout9 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=1, pop.size = 16, max.gene

m.out9 <- Match(Y = Y, Tr=Tr, X=X, M=1, Weight.matrix = genout9, caliper = c(0.5, 1e16, 1e16, 1e16,1e16

# -- 8. --
invisible(capture.output(genout10 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=2, pop.size = 16, max.gen

m.out10 <- Match(Y = Y, Tr=Tr, X=X, M=2, Weight.matrix = genout10, caliper = c(4, 1e16, 1e16, 4,1e16,1e

# -- 9. --
invisible(capture.output(genout11 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=3, pop.size = 16, max.gen

m.out11 <- Match(Y = Y, Tr=Tr, X=X, M=3, Weight.matrix = genout11, caliper = c(4, 1e16, 1e16, 4,1e16,1e

# -- 10. --
invisible(capture.output(genout12 <- GenMatch(Tr=Tr, X=X, estimand = "ATT", M=4, pop.size = 16, max.gen

m.out12 <- Match(Y = Y, Tr=Tr, X=X, M=4, Weight.matrix = genout12)


# we picked the best models by looking at the ks-statistics and p-value produced after balancing and ma
sprintf("Model 1: %f | Model 2: %f | Model 3: %f | Model 4: %f", m.out3$est, m.out4$est, m.out5$est, m.
```

```
## [1] "Model 1: -0.017361 | Model 2: -0.056090 | Model 3: 0.193894 | Model 4: 0.633013"
```

Our models show a high variability in treatment effects. However, all results are not far from zero. Together with the confidence intervals we obtained above, we are inclined to conclude that there is no statistically significant effect.

STEP 6 Repeat everything you've done for Steps 1-2, including the regression, genetic algorithm, code and estimating the treatment effect EXCEPT this time change the definition of treatment to cover 2 girls or more, and change the definition of control to cover 2 boys or more. Exclude all observations that don't meet these requirements. Be sure to explain (in a sentence or two) what you're doing with your new treatment and control definitions. Do your new definitions change anything?

When we define the treatment effect as at least 2 girls and no boys, we are increasing our dosage of treatment. Since we discard all treatment units that have less than 2 girls, we are making the treatment and control groups more diverse. With the increased treatment dosage and more diverse groups, it is easier to notice the impact of treatment, which is evident by the treatment effect we get now: 13 points in the total score.

Note: *Definition of the new treatment variable is as follows: Individuals in the treatment group should be having 2 or more girls and no boys, and individuals in the control group should be having 2 or more boys and no girls. What I had in mind was that such a situation increased the "dosage" of treatment vs. the "dosage" of control (and Rosenbaum alluded to this kind of observational design logic in one of the recently assigned articles). Note that you can't have the same units in the treatment group AND the control group – we should all know by now that such a situation would be wrong.*

```r
# Your code here
daughters2 <- daughters %>%
  filter((ngirls >= 2 & nboys==0) | (nboys >=2 & ngirls==0))

lm3 <- lm(nowtot ~ hasgirls + dems + repubs + christian + age + srvlng + demvote, data=daughters2)

# Treatment effect - coefficient for the treatment variable hasgirls
```

```
sed1 <- summary(lm3)$coefficients[, 2][2]
coefd1 <- lm3$coefficients[2][1]
lwrb1 <- coefd1 - sed1
uprb1 <- coefd1 + sed1

# Treatment effect - coefficient for the treatment variable hasgirls
sprintf("The Treatment Effect for hasgirls is %f, with 95%% CI [%f, %f]", coefd1, lwrb1, uprb1)
```

```
## [1] "The Treatment Effect for hasgirls is 12.292542, with 95% CI [8.791772, 15.793313]"
```

```
Y2 <- daughters2$nowtot
Tr2 <- daughters2$hasgirls
X2 <- cbind(daughters2$dems,daughters2$repubs, daughters2$christian, daughters2$age, daughters2$srvlng,

invisible(capture.output(genout22 <- GenMatch(Tr=Tr2, X=X2, estimand = "ATT", M=1, pop.size = 50, max.g

m.out22 <- Match(Y = Y2, Tr=Tr2, X=X2, M=1, Weight.matrix = genout22)

m.out22$est
```

```
##          [,1]
## [1,] 13.29787
```

STEP 7 It is NOT wise to match or balance on "totchi". What is the reason? Hint: You will have to look at what variables mean in the data set to be able to answer this question.

"totchi" stands for total children. Total children includes the number of boys and girls in the family. Hence, it also includes the treatment variable which is related to the number of girls. So, matching or balancing on total children is almost the same as matching or balancing on the treatment variable. Matching on treatment variable does not make sense because the point of matching is to have all covariates identical BESIDES the treatment variable. Balancing on treatment variable does not make sense because balancing presupposes that we have two groups - treatment and control. Every observation in treatment is already a treatment, and you cannot balance that.

QUESTION 3: COPD

Most causal studies on the health effects of smoking are observational studies (well, for very obvious reasons). In this exercise, we are specifically after answer the following question: Does smoking increase the risk of chronic obstructive pulmonary disease (COPD)? To learn more about the disease, read here: https://www.cdc.gov/copd/index.html

We'll use a sub-sample of the 2015 BRFSS survey (pronounced bur-fiss), which stands for Behavioral Risk Factor Surveillance System. The data is collected through a phone survey across American citizens regarding their health-related risk behaviors and chronic health conditions. Although, the entire survey has over 400,000 records and over 300 variables, we only sample 5,000 observations and 7 variables.

Let's load the data first and take a look at the first few rows:

```
brfss = read.csv("http://bit.ly/BRFSS_data") %>%
  clean_names()

head(brfss)
```

```
##   copd smoke    race   age    sex wtlbs avedrnk2
## 1   No     0   White 18-24 Female   180        4
## 2   No     0   White 55-64 Female   170        1
## 3   No     0   White   65+   Male   170        1
## 4    No      0 Hispanic 18-24   Male   185        1
## 5    No      0    White   65+ Female   150        1
## 6    No      0    White   65+   Male   180        2
```

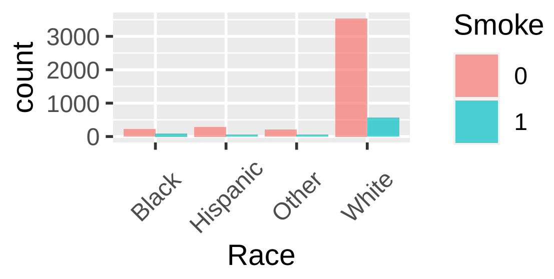
A summary of the variables is as follows:

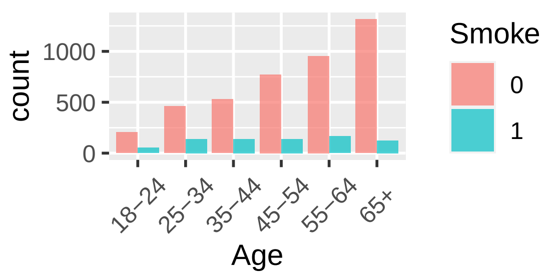
- copd: Ever told you have chronic obstructive pulmonary disease (COPD)?
- smoke: Adults who are current smokers (0 = no, 1 = yes)
- race: Race group
- age: age group
- sex: gender
- wtlbs: weight in pounds (lbs)
- avedrnk2: During the past 30 days, when you drank, how many drinks did you drink on average?

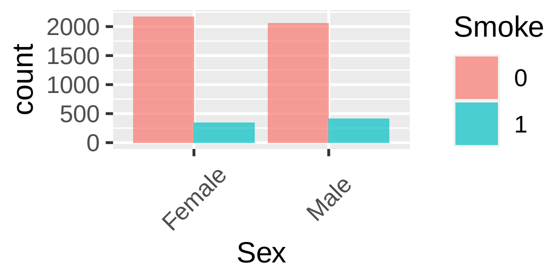
STEP 1 Check the balance of covariates before matching using any method of your choice. You can look at balance tables, balance plots, or love plots from any package of your choice. Do you see a balance across the covariates?

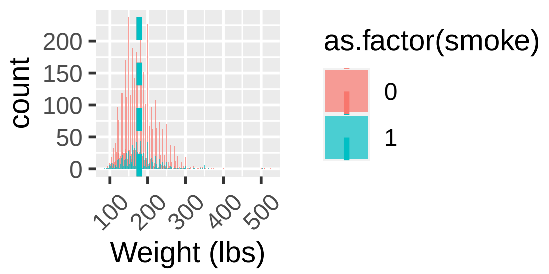
Note: *This is optional but you can use the* `gridExtra` *package and its* `grid.arrange()` *function to put all the 4 graphs in one 2 x 2 graph. Read more about the package and how to use it here: https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html. Set* `nrow = 2`*.*

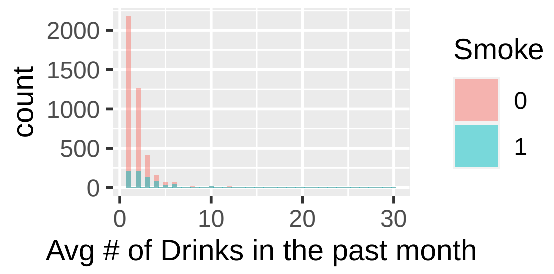
```
# Your code here
library(ggplot2)

brfss2 <- brfss %>% mutate(copd = ifelse(copd == "Yes", 1, 0))

dfstat2 <- ddply(brfss2, "smoke", summarise, wtlbs.mean=mean(wtlbs))

race_plot <- ggplot(brfss2, aes(x=as.factor(race), fill=as.factor(smoke))) + geom_histogram(stat="count
  labs(x="Race", fill = "Smoke") +
  theme(axis.text.x=element_text(size=9, angle=45, vjust=0.5))

age_plot <- ggplot(brfss2, aes(x=as.factor(age), fill=as.factor(smoke))) + geom_histogram(stat="count",
  labs(x="Age", fill = "Smoke") +
  theme(axis.text.x=element_text(size=9, angle=45, vjust=0.5))

sex_plot <- ggplot(brfss2, aes(x=as.factor(sex), fill=as.factor(smoke))) + geom_histogram(stat="count",
  labs(x="Sex", fill = "Smoke") +
  theme(axis.text.x=element_text(size=9, angle=45, vjust=0.5))

weight_plot <- ggplot(brfss2, aes(x=wtlbs, fill=as.factor(smoke))) +
  geom_histogram(binwidth=2, alpha=.7, position="identity") +
  geom_vline(data=dfstat2, aes(xintercept=wtlbs.mean,  colour=as.factor(smoke)),
             linetype="dashed", size=1) +
  labs(x="Weight (lbs)") +
  theme(axis.text.x=element_text(size=9, angle=45, vjust=0.5))

drink_plot <- ggplot(brfss2, aes(x=avedrnk2, fill=as.factor(smoke))) +
  geom_histogram(binwidth=.5, alpha=.5, position="identity") +
  labs(x="Avg # of Drinks in the past month", fill = "Smoke")
# for aesthetic reasons, we only plot 4 graphs because plotting more results in tiny graphs with low in
grid.arrange(race_plot, age_plot, sex_plot, weight_plot, drink_plot, nrow=3)
```











STEP 2 Now, let's do Mahalanobis distance matching. Note that you can use the same old `Match()` function. Use all covariates in the data set to match on, however, make sure you convert all categorical variables into factor variables (Google to see how). We are going to specify `estimand = "ATT"` in the `Match()` function. What's the treatment effect after matching?

```
# Your code here
brfss3 <- brfss %>%
  mutate(copd = as.factor(copd), smoke=as.numeric(smoke), race = as.factor(race), age=as.factor(age), s

sapply(brfss3, function(x) sum(is.na(x)))
```

```
##     copd    smoke     race      age      sex    wtlbs avedrnk2
##        0        0        0        0        0        0        0
```

```
X <- cbind(brfss3$race, brfss3$age, brfss3$sex, brfss3$wtlbs, brfss3$averdrnk2)
Y <- brfss3$copd
Tr <- brfss3$smoke

m.out1 <- Match(Y=Y, Tr=Tr, X=X, estimand = "ATT", Weight=2)

m.out1$est
```

```
##            [,1]
## [1,] 0.09261484
```

STEP 3 Provide a few sentences talking about the number of treated units dropped, and a few more sentences talking about the balance obtained.

From bal.tab and love.plot, we can see a good balance of covariates after matching. Additionally, we can look at the distribution of variale age before and after balance to see changes in the distribution. Since we used ATT - Average Treatment Effect for Treated, we have not dropped any Treatment units because all of them got matches from the control group. We can run m.out1$index.dropped to double check.

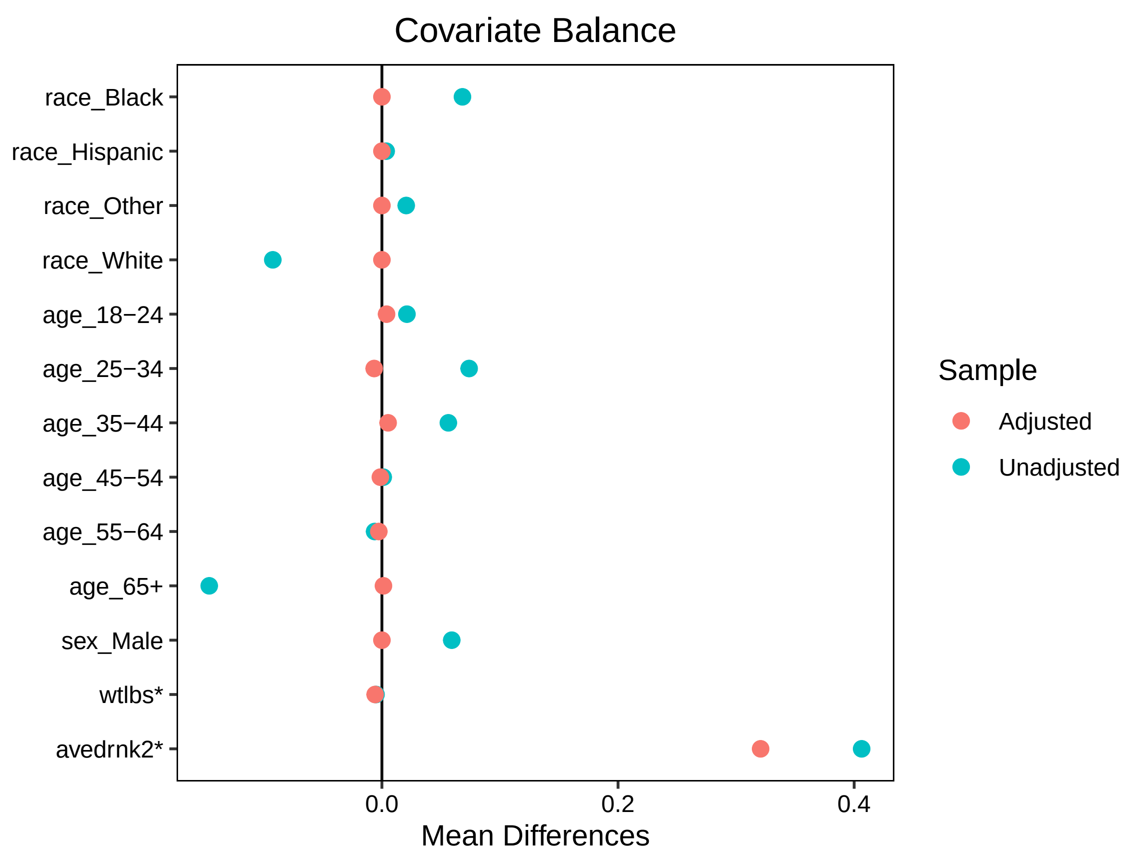
```
length(m.out1$index.dropped)
```

```
## [1] 0
```

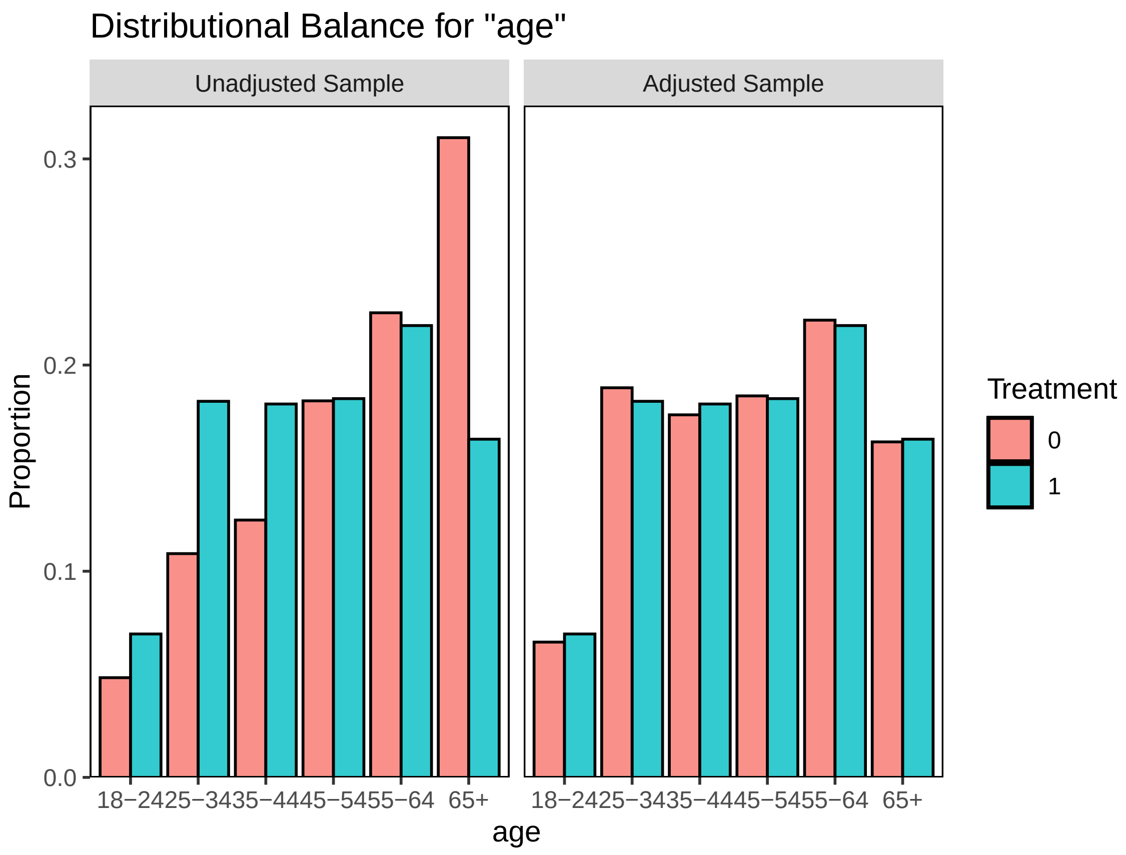
```
bal.tab(m.out1, formula = (smoke ~ race + age + sex + wtlbs + avedrnk2), data = brfss3)
```

```
## Balance Measures
##                    Type Diff.Adj
## race_Black       Binary   0.0000
## race_Hispanic    Binary   0.0000
## race_Other       Binary   0.0000
## race_White       Binary   0.0000
## age_18-24        Binary   0.0039
## age_25-34        Binary  -0.0066
## age_35-44        Binary   0.0052
## age_45-54        Binary  -0.0013
## age_55-64        Binary  -0.0026
## age_65+          Binary   0.0013
## sex_Male         Binary   0.0000
## wtlbs           Contin.  -0.0057
## avedrnk2        Contin.   0.3208
## 
## Sample sizes
##                      Control Treated
## All                  4238.       762
## Matched (ESS)         882.13     762
## Matched (Unweighted) 2602.       762
## Unmatched            1636.         0
```

```
love.plot(x=m.out1,  formula = (smoke ~ race + age + sex + wtlbs + avedrnk2), data = brfss3, method = "
```



```
bal.plot(x=m.out1, data=brfss, var.name = "age", which="both", formula = (smoke ~ race + age + sex + wtl
```



STEP 4 Now, let's do another Mahalanobis distance matching. Use all covariates in the data set in the propensity score estimation. However, this time make sure you specify `estimand = "ATE"` in the `Match()` function. What's the treatment effect after matching?

```r
# Your code here
m.out12 <- Match(Y=Y, Tr=Tr, X=X, estimand = "ATE", Weight=2)

m.out12$est
```

```
##              [,1]
## [1,] 0.1305759
```

STEP 5 Are your answers in Steps 2 and 4 different? Why? What does the matching process do differently in each step? Which answer do you trust more?

ATT vs. ATE

In Step 4, because of using ATE, we got a larger treatment effect: 0.131 vs. 0.092.

When we find ATT, we find a match for every Treatment unit and estimate the treatment effect for every subject in the treated group. Since we prioritize the Treatment unit, it means that not every Control observation will contribute to the computation of treatment effect. This gives us a less holistic picture and a smaller variance in the control sample. However, it may be the case the the unmatched control units were outliers that would only skew our treatment effect.

For ATE, we first find a match for every Treatment unit, then for every Control unit, then we find the treatment effect for each matched units and average out the effect. ATE finds matches for both Treated and Control units, thus increasing variance in the sample size used for calculating the treatment effect. However, if we don't impose reasonable constraints (such as with caliper), some outlier Control units might be forced to be matched, thus skewing our results.

At the end, ATE's offers a more comprehensive approach by finding matches for both groups and calculating the average treatment effect, resulting in a higher variance of the sample size and a more trustworthy treatment effect.

BONUS QUESTION: Sensitivity Analysis

STEP 1 Use the BRFSS data set from the previous question. Now, identify the critical value of Gamma as we discussed in the class. Do it using rbounds: https://nbviewer.jupyter.org/gist/viniciusmss/a156c3f22081fb5c690cdd58658f61fa

```
# Your code here
library(rbounds)

psens(m.out1, Gamma=4, GammaInc=.1)$bounds[c(20:40), ]
```

```
##       Gamma Lower bound Upper bound
## 20      2.9           0      0.0001
## 21      3.0           0      0.0003
## 22      3.1           0      0.0012
## 23      3.2           0      0.0042
## 24      3.3           0      0.0122
## 25      3.4           0      0.0301
## 26      3.5           0      0.0646
## 27      3.6           0      0.1217
## 28      3.7           0      0.2047
## 29      3.8           0      0.3110
## 30      3.9           0      0.4326
## 31      4.0           0      0.5577
## NA       NA          NA          NA
## NA.1     NA          NA          NA
## NA.2     NA          NA          NA
## NA.3     NA          NA          NA
## NA.4     NA          NA          NA
## NA.5     NA          NA          NA
## NA.6     NA          NA          NA
## NA.7     NA          NA          NA
## NA.8     NA          NA          NA
```

STEP 2 Then, write a paragraph explaining what you found. Your paragraph should include numbers obtained from your analysis and explain what those numbers mean as simply as you can.

Gamma is how many times a unit is more likely to end up in the treatment group rather than control. When Gamma is 1, the probability is the same. When Gamma is 3, a unit is 3 times more likely to end up in the treatment group. This probability is influenced by the difference in unobserved confounders that might influence the assignment mechanism and the outcome. From the table above, we see that out experiment's results would become insignificant (p-value > 0.05) at Gamme $= 3.5$. This means that even if a unit is 3.4 times more likely to end up in the treatment group (implying an imbalance in unobserved confounders' distribution), our results would still be statistically significant. While in practice we don't know whether we have any unobserved confounding variables, the Sensitivity Analysis tells us how robust our results are. Within the context of medical research of impact of smoking on the lung disease, Gamma of 3.5 implies that our results are robust.

End of Assignment

Final Steps

Before finalizing your project you'll want be sure there are **comments in your code chunks** and **text outside of your code chunks** to explain what you're doing in each code chunk. These explanations are incredibly helpful for someone who doesn't code or someone unfamiliar to your project.

You have two options for submission:

1. You can complete this .rmd file, knit it to pdf and submit the resulting .pdf file on Forum.
2. You can complete the Google Doc version of this assignment, include your code, graphs, results, and your explanations wherever necessary and download the Google Doc as a pdf file and submit the pdf file on Forum. If you choose this method, you need to make sure you will provide a link to an .R script file where you code can be found (you can host your code on Github or Google Drive). Note that links to Google Docs are not accepted as your final submission.

Knitting your R Markdown Document

Last but not least, you'll want to **Knit your .Rmd document into a pdf document**. If you get an error, take a look at what the error says and edit your .Rmd document. Then, try to Knit again! Troubleshooting these error messages will teach you a lot about coding in R. If you get any error that doesn't make sense to you, post it on Perusall.

Good Luck! The CS112 Teaching Team