Introduction to Causal Inference

Solutions to Problem Set 3

Professor: Teppei Yamamoto

Due Monday, July 18 (at beginning of class)

Only the required problems are due on the above date. The optional problems will not directly count toward your grade, though you are encouraged to complete them as your time permits. If you choose not to work on the optional problems, use them for your self-study during the summer vacation.

Required Problems

Problem 1

After months of deep thought, you decide that your dissertation will focus on the relationship between X and Y. In preparation for the meeting with your committee, you read all the relevant literature, conduct field interviews, and finally write down a directed acyclic graph that you believe captures all of the relevant variables, and their inter-relationships. You proudly present Figure 1 to your committee.

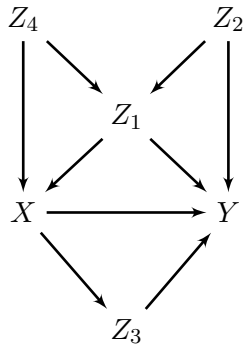


Figure 1: Your Dissertation In A Graph

(a) Your committee chair, who is not used to DAGs, asks you to explain the purpose of your DAG. As a researcher, what does it buy you, and what does it not buy you?

The purpose of the DAG is to explicitly state our beliefs about the relationships of variables. In this particular setting, the DAG is helpful in that it helps us to understand the conditions under which we might identify a causal effect. Of course, the DAG is only as helpful as it is

accurate – if we get the directionality of various relationships wrong, or if we fail to include certain variables, the DAG may not help at all, and in fact may induce us to make poor choices. It is always better to start with a solid/plausible research design to motivate the particular DAG we draw.

(b) Explain, in your own words, Pearl's back-door criterion.

Pearl's back-door criterion is a way to rule out confounding via conditioning, thus identifying the effect of one variable on another. For identifying the effect of any X on any Y, the back-door criterion has two parts. First, the conditioning set $\mathbb{S}$ may not include any descendent of X. That means, anything that X affects cannot be conditioned on (c.f. post-treatment bias). Second, the conditioning set must block all back-door paths from X to Y. That is, by conditioning on our set, we should break any back-door paths that may simultaneously generate confounded covariance in X and Y.

(c) For Figure 1, enumerate all paths from X to Y.

$\{X \rightarrow Y\}$
$\{X \rightarrow Z_3 \rightarrow Y\}$
$\{X \leftarrow Z_1 \rightarrow Y\}$
$\{X \leftarrow Z_1 \leftarrow Z_2 \rightarrow Y\}$
$\{X \leftarrow Z_4 \rightarrow Z_1 \rightarrow Y\}$
$\{X \leftarrow Z_4 \rightarrow Z_1 \leftarrow Z_2 \rightarrow Y\}$

(d) In the path $\{X \leftarrow Z_4 \rightarrow Z_1 \leftarrow Z_2 \rightarrow Y\}$, what type of node is Z_1? Does conditioning on Z_1 block or unblock this path from X to Y?

Z_1 is a collider or "collision node". That means that it is simultaneously the child of two variables, and a parent of none. Without conditioning on Z_1, Z_1 blocks the path $\{X \leftarrow Z_4 \rightarrow Z_1 \leftarrow Z_2 \rightarrow Y\}$. Conditioning on Z_1 would unblock the path.

(e) Now consider path $\{X \leftarrow Z_1 \rightarrow Y\}$. In this path, what type of node is Z_1? Does conditioning on Z_1 here block or unblock this path?

Z_1 is now a parent of both X and Y (you could call it an "arrow emitting node"). This means it jointly affects both X and Y, so it is essentially a potential confounder in the usual sense. Failing to condition on it will open a back-door path between X and Y, but conditioning on it will block the path.

(f) Previous research in your area has exclusively conditioned on Z_1, claiming that this is necessary and sufficient to identify the relationship between X and Y. Given Figure 1, does conditioning on Z_1 satisfy the back-door criterion for identifying the effect of X on Y? Why or why not?

No, this does not satisfy the back-door criterion. By conditioning on the collider Z_1, the backdoor path $\{X \leftarrow Z_4 \rightarrow Z_1 \leftarrow Z_2 \rightarrow Y\}$ is unblocked.

(g) Based on your DAG, enumerate the minimum conditioning sets that satisfy the back-door criterion for identifying the effect of X on Y.

$\{Z_1, Z_4\}$; $\{Z_1, Z_2\}$. Conditioning on either of these sets (note they are minimum) will satisfy the back-door criterion.

(h) **(Optional)** Consider the DAG in Figure 2. To identify the effect of X on Y, should we include M in our conditioning set? Why or why not? What is this an example of?

Now, use `R` to demonstrate your answer by simulation. Repeatedly simulate data from a data generating process consistent with the DAG in Figure 2. You may choose particular parameters, values, or distributions for each variable as you so wish, as long as the causal relationships between the variables match those encoded in the DAG. Then, for each simulated sample from the DGP, estimate the ATE of X on Y both with and without conditioning on M. Constrast your results graphically, and interpret whether they support your answer above.

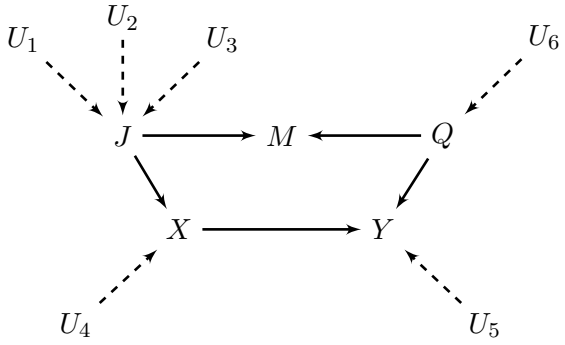
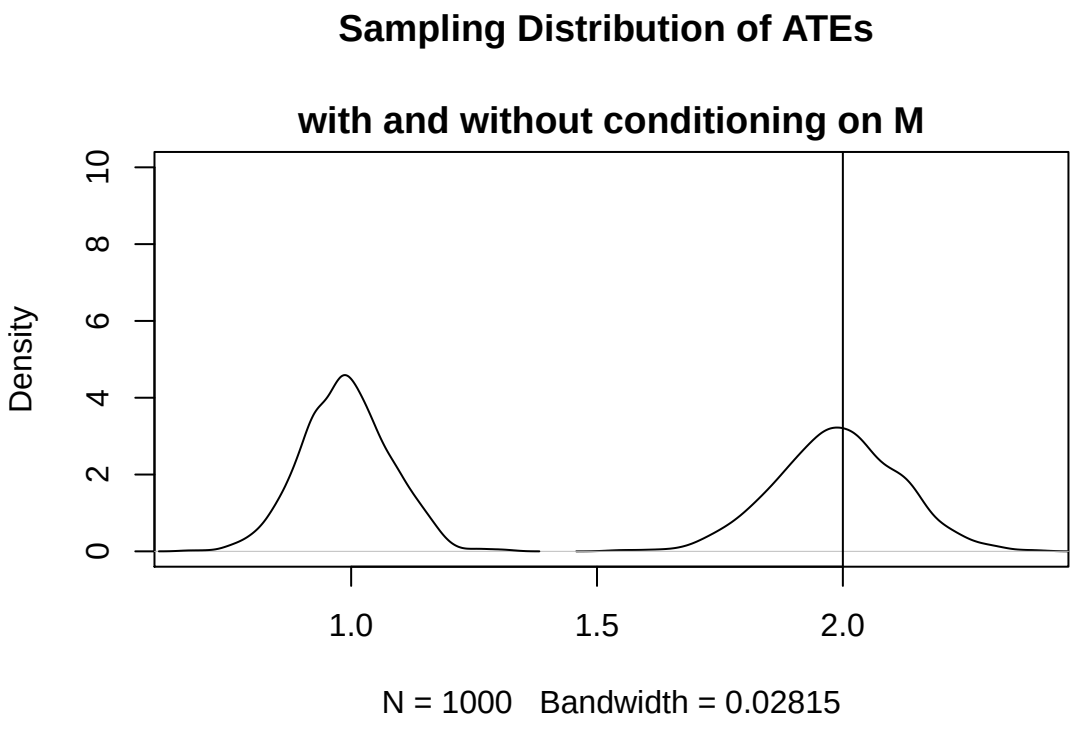


Figure 2: Directed Acyclic Reindeer

We should most definitely not condition on M. This would be a case of M-bias. As we can see from our simulations our estimates of the ATE when we condition on M are biased.

```
###Start by generating our variables:
#Choose some parameters, these will be what we estimate
alpha = 1.2
beta = 2
gamma = 4.5
#set sims:
sims = 1000
#holdes
beta_hat = c()
beta_m_hat = c()
for(i in 1:sims){
##To do this, follow the logic of the DAG:
```

```r
J = rnorm(1000,0,5) + runif(1000,0,1) + rbinom(1000,1,.5)
#Make X a function of J
X = .5*J + rnorm(1000,0,5)
#Create Q
Q = rnorm(1000,10,5)
#Generate Y
Y = alpha + beta*X + gamma*Q + rnorm(1000,0,1)
#Create M
M = J + Q
beta_hat[i] = lm(Y ~ X)$coefficient[2]
beta_m_hat[i] = lm(Y ~ X + M)$coefficient[2]
}

plot(density(beta_hat), xlim=c(min(beta_m_hat),max(beta_hat)),
     main="Sampling Distribution of ATEs
     \nwith and without conditioning on M",ylim=c(0,10))
lines(density(beta_m_hat))
abline(v=beta)
```



Problem 2

In this problem you will analyze the data in `child_soldiering.csv`, from the Blattman and Annan (2010) article "The Consequences of Child Soldiering."[1] The authors are interested in the impact of abduction by the Lord's Resistance Army on political, economic, and psychological outcomes; the data are from a survey of male youth in war-afflicted regions of Uganda. We will focus on the effect of abduction, which appears in the data as `abd`, on years of education, `educ`. Other variables in the data are:

- `C.ach`, `C.akw`, etc.: sub-district identifiers
- `age`: respondent's age in years
- `fthr.ed`: father's education (years)
- `mthr.ed`: mother's education (years)
- `orphan96`: indicator for whether parents died before 1997
- `fthr.frm`: indicator for whether father is a farmer
- `hh.size96`: household size in 1996

(a) Check covariate balance in the unmatched dataset using the `Matching` package's `MatchBalance()` function, for all covariates. Your output should be in the form of a balance table. Make sure to present statistical tests of the equality of means and equality of distributions. Based on your table, which of the observed covariates seem to be the most important factors in selection into abduction? Using the (naïve) difference-in-means estimator, estimate the ATE of abduction on education without adjusting for any of the covariates, and report your estimate along with a standard error.

```
library(Matching)
library(ebal)
library(xtable)
d = read.csv('child_soldiering.csv')
 # Estimate the ATE as the difference in means
ate = mean(d$educ[d$abd == 1]) - mean(d$educ[d$abd == 0])
 # And estimate its standard error
ate.se = sqrt(var(d$educ[d$abd == 1])/length(d$educ[d$abd == 1]) +
                var(d$educ[d$abd == 0])/length(d$educ[d$abd == 0]))
 # Check balance in unmatched data
bal.formula = formula(abd ~ C.ach + C.akw + C.ata + C.kma + C.oro
                      + C.pad + C.paj + C.pal + age +fthr.ed + mthr.ed
```

[1] Reading the article is not necessary for completing the problem, but can be found at http://www.chrisblattman.com/documents/research/2010.Consequences.RESTAT.pdf.

```
                    + orphan96 + fthr.frm + hh.size96 + educ)
 # Print a balance table
mb.unmatched = MatchBalance(bal.formula, data = d, print.level=0)
tab.unmatched = baltest.collect(mb.unmatched,
                                var.names = colnames(d)[-1], after=F)
print(xtable(tab.unmatched[, 1:7],  label='tab:unmatched-bal',
             caption='Covariate Balance in Unmatched Data'),
      caption.placement='top')
```

Table 1: Covariate Balance in Unmatched Data

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.15	0.11	10.80	11.44	1.28	0.13	
C.akw	0.16	0.08	21.68	24.65	1.83	0.00	
C.ata	0.10	0.20	-32.55	-27.67	0.57	0.00	
C.kma	0.15	0.12	9.26	9.73	1.23	0.19	
C.oro	0.05	0.14	-37.92	-29.12	0.42	0.00	
C.pad	0.12	0.12	-0.20	-0.20	0.99	0.98	
C.paj	0.15	0.10	13.25	14.27	1.38	0.06	
C.pal	0.11	0.13	-5.21	-5.05	0.89	0.51	
age	21.37	20.15	24.24	24.49	1.04	0.00	0.01
fthr.ed	5.76	6.07	-8.59	-8.46	0.94	0.27	0.83
mthr.ed	2.09	2.49	-14.49	-13.31	0.73	0.09	0.37
orphan96	0.08	0.08	0.99	1.00	1.03	0.90	
fthr.frm	0.90	0.91	-3.83	-3.94	1.12	0.60	
hh.size96	8.09	8.70	-15.50	-14.62	0.80	0.06	0.05
educ	6.82	7.42	-21.34	-20.51	0.86	0.01	0.06

Table 1 shows that treated and control units look alike on a number of covariates but quite different with respect to age and some of the location variables; this suggests that these variables are predictive of treatment. The naive ATE, estimated as the difference in means, is -0.5954243; its standard error is 0.222192.

(b) Consider the authors' description of abduction:

> Abduction was large-scale and seemingly indiscriminate; 60,000 to 80,000 youth are estimated to have been abducted and more than a quarter of males currently aged 14 to 30 in our study region were abducted for at least two weeks. Most were abducted after 1996 and from one of the Acholi districts of Gulu, Kitgum, and Pader.
>
> Youth were typically taken by roving groups of 10 to 20 rebels during night raids on rural homes. Adolescent males appear to have been the most pliable, reliable and effective forced recruits, and so were disproportionately targeted by the LRA.

Youth under age 11 and over 24 tended to be avoided and had a high probability of immediate release.

Given this description, choose some covariates on which to match, and then do so using the `Match()` function. Be sure to carefully check the options available to you in this function. For now, use only one match for each treated unit, use the Mahalanobis distance metric to determine weights, and do not (yet) use exact matching. Apply the matching estimator to estimate the average effect of abduction on education among those who were abducted, i.e., the ATT. Report your estimate and standard error, as well as balance statistics for the matched data.

```
# Match on a set of covariates
vars = c('age', 'fthr.ed', 'mthr.ed', 'orphan96', 'fthr.frm', 'hh.size96')
# Function defaults will estimate the ATT with M=1
m.first = Match(Y=d$educ, Tr=d$abd, X = d[, vars], Weight=2)
summary(m.first)
```

```
##
## Estimate...  -0.50152
## AI SE......  0.26428
## T-stat.....  -1.8976
## p.val......  0.057744
##
## Original number of observations..............  741
## Original number of treated obs...............  462
## Matched number of observations...............  462
## Matched number of observations  (unweighted).  521
```

```
# Check balance
mb.matched = MatchBalance(bal.formula, data=d, match.out=m.first, print.level=0)
# ATT and its SE can be found in the est and se elements of m.first
# Print a balance table
tab.matched = baltest.collect(mb.matched, var.names = colnames(d)[-1])
print(xtable(tab.matched[, 1:7], caption='Covariate Balance in Matched Data',
             label='tab:matched-bal'), caption.placement='top')
```

The ideal set of covariates for matching is debatable. Here, we choose not to match on the location indicators, on the grounds that it would be better to have pairs from different villages but who share other characteristics, than to have pairs from the same village but with worse balance on other characteristics. The estimate of the ATT is -0.5015152 and its standard

Table 2: Covariate Balance in Matched Data

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.15	0.10	14.89	14.89	1.45	0.01	
C.akw	0.16	0.10	16.30	16.30	1.50	0.00	
C.ata	0.10	0.22	-40.73	-40.73	0.52	0.00	
C.kma	0.15	0.12	10.01	10.01	1.26	0.11	
C.oro	0.05	0.14	-38.45	-38.45	0.42	0.00	
C.pad	0.12	0.09	9.94	9.94	1.32	0.12	
C.paj	0.15	0.12	9.91	9.91	1.25	0.10	
C.pal	0.11	0.12	-3.08	-3.08	0.93	0.65	
age	21.37	21.13	4.64	4.64	1.07	0.01	0.63
fthr.ed	5.76	5.68	2.32	2.32	1.09	0.17	0.95
mthr.ed	2.09	2.07	0.86	0.86	1.00	0.49	1.00
orphan96	0.08	0.08	0.00	0.00	1.00	1.00	
fthr.frm	0.90	0.90	0.00	0.00	1.00	1.00	
hh.size96	8.09	8.00	2.23	2.23	1.19	0.19	0.24
educ	6.82	7.32	-17.97	-17.97	0.80	0.01	0.01

error is 0.2642838. As you can see from the balance table, the balance improves once we match. But always remember that the curse of dimensionality means adding more covariates may make matching more and more difficult.

(c) Re-estimate the ATT using exact matching on `C.ach, C.akw, C.ata, C.oro, C.paj`, and `age`. Report your estimate, its standard error, and produce a balance table as before. (**Optionally**, also show a balance plot that compares the differences between treated and control for this match and the match in (b), similar to the plot shown in slide 40 of the lecture slides.) In general, do your results differ from previous results? Why or why not?

```
# Match exactly on the geo IDs and age
m.exact = Match(Y=d$educ, Tr=d$abd, X = d[, 2:10], exact=TRUE)
summary(m.exact)

##
## Estimate...  -0.86631
## AI SE......  0.21474
## T-stat.....  -4.0342
## p.val......  5.4794e-05
##
## Original number of observations..............  741
## Original number of treated obs...............  462
## Matched number of observations...............  371
## Matched number of observations  (unweighted).  961
##
```

```
## Number of obs dropped by 'exact' or 'caliper'  91
```

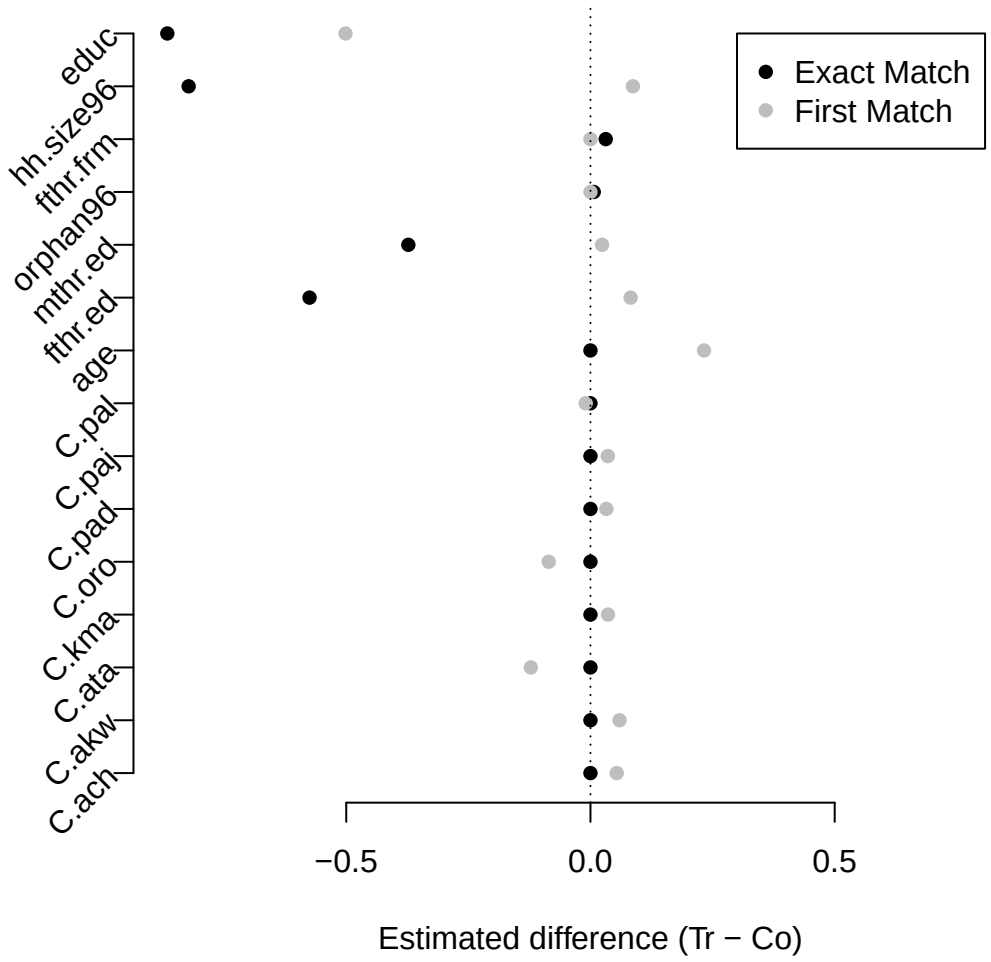
```
# Check balance
mb.exact = MatchBalance(bal.formula, data=d, match.out=m.exact, print.level=0)
# ATT and its SE can be found in the est and se elements of m.exact
# Print the table
tab.exact = baltest.collect(mb.exact, var.names = colnames(d)[-1])
print(xtable(tab.exact[, 1:7], caption='Covariate Balance in Exactly Matched Data',
             label='tab:exact-bal'), caption.placement='top')
```

Table 3: Covariate Balance in Exactly Matched Data

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.17	0.17	0.00	0.00	1.00	1.00	
C.akw	0.13	0.13	0.00	0.00	1.00	1.00	
C.ata	0.10	0.10	0.00	0.00	1.00	1.00	
C.kma	0.14	0.14	0.00	0.00	1.00	1.00	
C.oro	0.06	0.06	0.00	0.00	1.00	1.00	
C.pad	0.13	0.13	0.00	0.00	1.00	1.00	
C.paj	0.16	0.16	0.00	0.00	1.00	1.00	
C.pal	0.11	0.11	0.00	0.00	1.00	1.00	
age	20.82	20.82	0.00	0.00	1.00	1.00	1.00
fthr.ed	5.61	6.18	-16.37	-16.37	0.89	0.03	0.36
mthr.ed	2.15	2.52	-13.26	-13.26	0.74	0.09	0.37
orphan96	0.08	0.07	2.55	2.55	1.09	0.72	
fthr.frm	0.91	0.88	10.82	10.82	0.77	0.16	
hh.size96	8.06	8.88	-22.27	-22.27	0.68	0.01	0.00
educ	6.79	7.65	-32.12	-32.12	0.88	0.00	0.00

```
#make a balance plot:
 bal.var = colnames(d)[-1]
 cov.diff1=c()
 for(i in 1:length(bal.var)){
 cov.diff1[i] = mb.matched$AfterMatching[[i]]$tt$estimate
 }
 cov.diff=c()
 for(i in 1:length(bal.var)){
 cov.diff[i] =mb.exact$AfterMatching[[i]]$tt$estimate
 }

 plot(cov.diff, c(1:15), axes=F, ylab="", xlab="Estimated difference (Tr - Co)",
      pch=16, xlim=c(-max(abs(cov.diff)),max(abs(cov.diff))))
```

```
points(cov.diff1,c(1:15), pch=16, col="grey")
abline(v=0,lty=3)
axis(1)
axis(2, at=c(1:15), labels=F)
text(y = c(1:15), par("usr")[1], labels = bal.var, srt = 45, pos = 2, xpd = TRUE)
legend(0.3, 15, pch=c(16,16), col=c("black", "grey"),
       legend=c("Exact Match", "First Match"))
```



After matching exactly on the location indicators and age, the estimate of the ATT is −0.8663073 (substantially different) and the standard error is 0.2147419. Balance on the exactly matched variables is perfect, of course, but much worse on other variables. The location indicators are binary variables, and each observation records a 1 for only one of the location indicators and 0 for all the others, so finding exact matches for each observation on

all the location indicators is easy. All we are doing is finding pairs of people of the same age from the same village, one of whom was abducted and one of whom was not. It is easy to imagine two such people would differ on a range of other characteristics.

(d) Re-estimate the ATT as in (b), but using one-to-many matching. That is, for each treated unit, choose $M = 2$ and $M = 10$ control matches, where M is the number of matches for each treated unit. Compare the results with the estimates obtained in (b). Do they differ? Why? What trade-offs are we making by increasing the number of matches?

```
# Match with M=2
m.m2 = Match(Y=d$educ, Tr=d$abd, X = d[, vars], M=2)
summary(m.m2)

##
## Estimate...  -0.57937
## AI SE......  0.23427
## T-stat.....  -2.4731
## p.val......  0.013396
##
## Original number of observations..............  741
## Original number of treated obs...............  462
## Matched number of observations...............  462
## Matched number of observations  (unweighted).  1041
```

```
# Check balance
mb.m2 = MatchBalance(bal.formula, data=d, match.out=m.m2, print.level=0)
# Print a balance table
tab.m2 = baltest.collect(mb.m2, var.names = colnames(d)[-1])
print(xtable(tab.m2[, 1:7], caption='Covariate Balance in Matched, M=2',
             label='tab:m2-bal'), caption.placement='top')
```

```
# ATT and its SE can be found in the est and se elements of m.m2
```

With $M = 2$, we get a point estimate of -0.5793651 and a standard error of 0.2342696.

```
# Repeat with M=10
m.m10 = Match(Y=d$educ, Tr=d$abd, X = d[, vars], M=10)
mb.m10 = MatchBalance(bal.formula, data=d, match.out=m.m10, print.level=0)
tab.m10 = baltest.collect(mb.m10, var.names = colnames(d)[-1])
print(xtable(tab.m10[, 1:7], caption='Covariate Balance in Matched, M=10',
             label='tab:m10-bal'), caption.placement='top')
```

Table 4: Covariate Balance in Matched, M=2

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.15	0.11	12.57	12.57	1.35	0.04	
C.akw	0.16	0.09	18.70	18.70	1.63	0.00	
C.ata	0.10	0.22	-41.57	-41.57	0.52	0.00	
C.kma	0.15	0.12	8.60	8.60	1.21	0.17	
C.oro	0.05	0.13	-35.97	-35.97	0.43	0.00	
C.pad	0.12	0.09	8.78	8.78	1.27	0.17	
C.paj	0.15	0.09	16.72	16.72	1.55	0.00	
C.pal	0.11	0.14	-9.09	-9.09	0.82	0.19	
age	21.37	21.16	4.06	4.06	1.04	0.04	0.41
fthr.ed	5.76	5.63	3.92	3.92	1.10	0.04	0.26
mthr.ed	2.09	2.07	0.74	0.74	1.02	0.63	0.96
orphan96	0.08	0.08	0.00	0.00	1.00	1.00	
fthr.frm	0.90	0.90	0.00	0.00	1.00	1.00	
hh.size96	8.09	7.99	2.53	2.53	1.19	0.24	0.05
educ	6.82	7.40	-20.76	-20.76	0.82	0.00	0.00

With $M = 10$, we get a point estimate of -0.4867904 and a standard error of 0.2240147.

In general, as we set M larger, we average over more control units to estimate the non-treatment outcome for each treated unit, with good results for precision. Note that the standard error shrinks from $M = 1$ to $M = 2$ and to $M = 10$. This comes at a cost, however. Earlier we picked only the best match for each treated unit; now we pick the best match and other worse matches. This may well worsen balance between the treatment and control groups, especially at $M = 10$. (Your estimated ATTs may nonetheless be similar to those obtained earlier.)

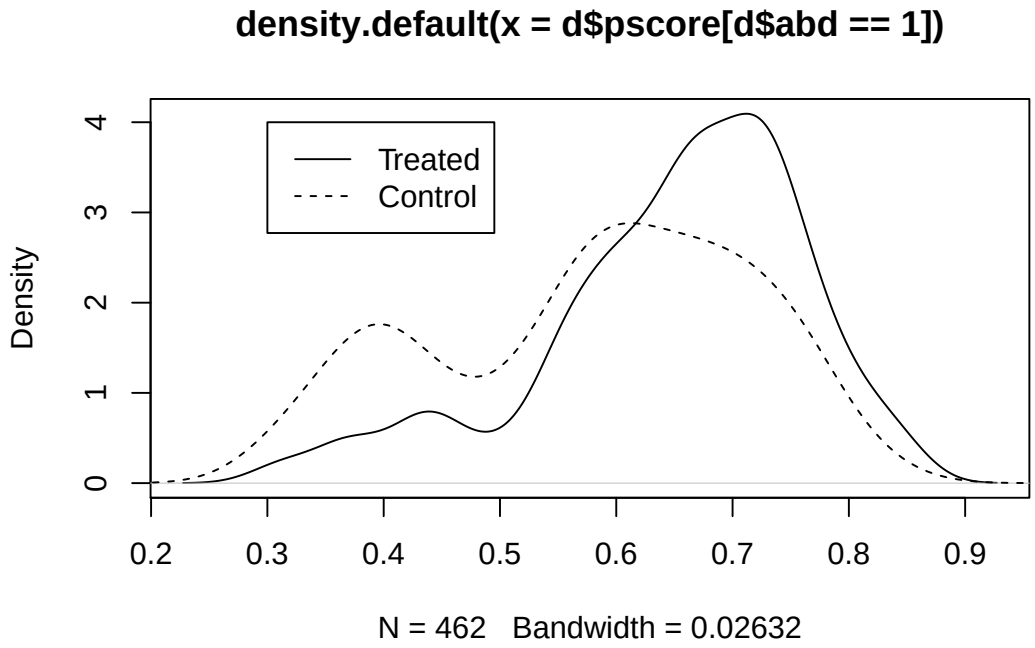
(e) Now, instead of matching on covariates, we'll use propensity scores to estimate the ATT.

i) Using logit, regress `abd` on `C.ach, C.akw, C.ata, C.oro, C.paj`, and `age`. For each unit, generate a propensity score as the predicted probability of taking treatment. On one plot, overlay the density curves of the propensity scores for the treated and untreated units. What does this plot reveal?

```
# Generate the pscore
ps.model = glm(abd ~ C.ach + C.akw + C.ata + C.oro + C.paj + age,
               data=d, family = binomial(link=logit))
d$pscore = predict(ps.model, type="response")
plot(density(d$pscore[d$abd==1]))
lines(density(d$pscore[d$abd==0]), lty=2)
legend(0.3,4, lty=c(1,2),legend = c("Treated", "Control"))
```

Table 5: Covariate Balance in Matched, M=10

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.15	0.11	11.59	11.59	1.31	0.06	
C.akw	0.16	0.07	24.00	24.00	2.03	0.00	
C.ata	0.10	0.21	-36.83	-36.83	0.54	0.00	
C.kma	0.15	0.11	12.93	12.93	1.37	0.03	
C.oro	0.05	0.14	-39.08	-39.08	0.41	0.00	
C.pad	0.12	0.12	-1.04	-1.04	0.98	0.88	
C.paj	0.15	0.10	13.99	13.99	1.41	0.02	
C.pal	0.11	0.14	-8.07	-8.07	0.84	0.24	
age	21.37	20.79	11.46	11.46	1.12	0.00	0.00
fthr.ed	5.76	5.71	1.55	1.55	1.24	0.58	0.00
mthr.ed	2.09	1.90	6.96	6.96	1.17	0.01	0.04
orphan96	0.08	0.08	0.97	0.97	1.03	0.27	
fthr.frm	0.90	0.91	-2.25	-2.25	1.07	0.09	
hh.size96	8.09	7.84	6.36	6.36	1.47	0.05	0.00
educ	6.82	7.31	-17.44	-17.44	0.89	0.01	0.00



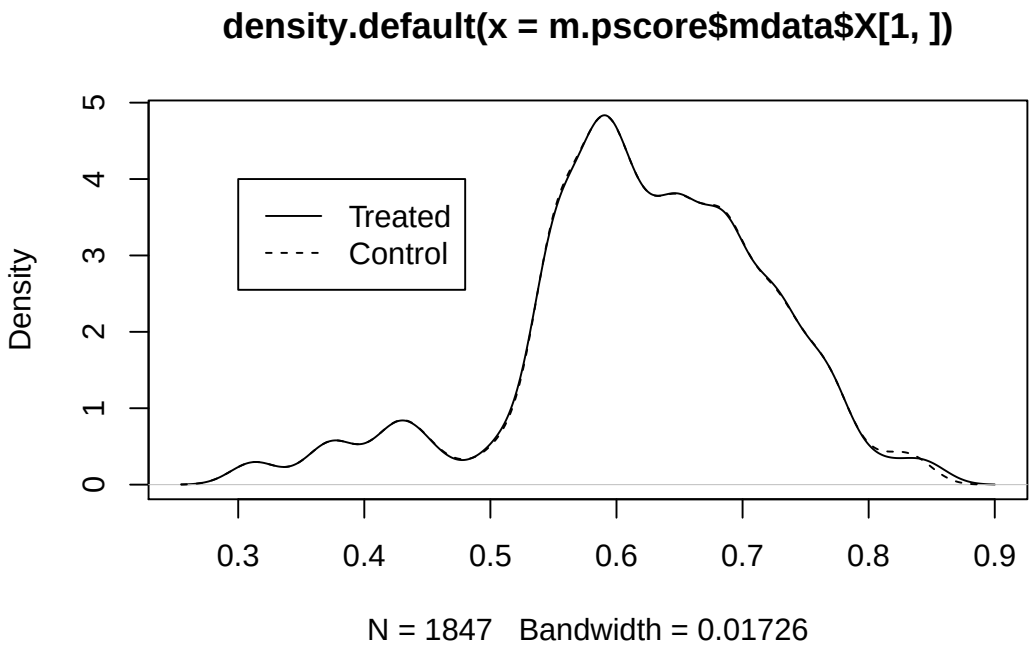
The plot reveals that the propensity for treatment, at least as predicted by the logit specification we chose, tends to be higher among treated units (shown by the solid line) that in control units (the dashed line). This is in fact by construction, as we are using the observed treatment indicator to estimate the propensity scores. If, prior to any matching or adjustment, treated and control units were perfectly balanced on the covariates we included in our specification, the densities of the estimated propensities would be near identical. It is of course possible that, even in such a situation, differences

in the densities of propensity scores might still result from sampling error.

ii) Using one-to-one matching on propensity scores, estimate the ATT, its standard error, and plot the densities of the propensity scores for the treated and untreated units. Assess balance using a balance table, as before.

```
  ##Matching:
  m.pscore <- Match(Y = d$educ, Tr = d$abd, X = d$pscore)
  summary(m.pscore)

##
## Estimate...  -0.96162
## AI SE......  0.27455
## T-stat.....  -3.5025
## p.val......  0.00046094
##
## Original number of observations..............  741
## Original number of treated obs...............  462
## Matched number of observations...............  462
## Matched number of observations  (unweighted).  1847
```

```
  #Plot the p-scores after matching
  plot(density(m.pscore$mdata$X[1,]))
  lines(density(m.pscore$mdata$X[2,]), lty=2)
  legend(0.3,4, lty=c(1,2),legend = c("Treated", "Control"))
```




```
mb.pscore <- MatchBalance(bal.formula, data = d, match.out = m.pscore,
                          print.level = 0)
# ATT and its SE can be found in the est and se elements of
# Print
tab.pscore <- baltest.collect(mb.pscore,
                              var.names = colnames(d)[-c(1,17)])
print(xtable(tab.pscore[, 1:7], caption = "Covariate Balance
             in Propensity Score Matched Data",
             label = "tab:pscore-bal"), caption.placement = "top")
```

Using one-to-one matching on propensity scores, we retrieve a point estimate of -0.9616213 and a standard error of 0.2745541. This is a larger and slightly less certain point estimate than those from the earlier approaches. Now the plot shows that the propensity scores are very balanced across treated and control (unsurprisingly). The table shows good balance on the covariates we included in the propensity score model.

iii) Now estimate the ATT using inverse-probability-weighting (IPW).

We can estimate the ATT using IPW as follows:

$$\tau_{\hat{ATE}} = \frac{1}{N_1} \sum_{i=1}^{N} \left\{ Y_i \cdot \frac{D_i - pscore(X_i)}{1 - pscore(X_i)} \right\}$$

Table 6: Covariate Balance in Propensity Score Matched Data

	mean.Tr	mean.Co	sdiff	sdiff.pooled	var.ratio	T pval	KS pval
C.ach	0.15	0.15	0.00	0.00	1.00	1.00	
C.akw	0.16	0.15	1.19	1.19	1.02	0.67	
C.ata	0.10	0.09	4.33	4.33	1.13	0.06	
C.kma	0.15	0.12	8.30	8.30	1.20	0.13	
C.oro	0.05	0.06	-3.90	-3.90	0.87	0.16	
C.pad	0.12	0.15	-7.78	-7.78	0.85	0.22	
C.paj	0.15	0.16	-1.21	-1.21	0.98	0.67	
C.pal	0.11	0.12	-2.75	-2.75	0.94	0.65	
age	21.37	21.34	0.47	0.47	1.00	0.78	0.84
fthr.ed	5.76	6.45	-19.41	-19.41	0.82	0.00	0.41
mthr.ed	2.09	2.45	-12.94	-12.94	0.70	0.07	0.04
orphan96	0.08	0.07	2.80	2.80	1.10	0.66	
fthr.frm	0.90	0.88	6.30	6.30	0.86	0.34	
hh.size96	8.09	9.55	-37.27	-37.27	0.47	0.00	0.00
educ	6.82	7.78	-34.46	-34.46	0.82	0.00	0.00

```
##Weighting:
#ATE:
#sum(d£educ * (d£abd-d£pscore)/(d£pscore*(1-d£pscore)))/ nrow(d)
#ATT:
ipw.att = sum(d$educ * (d$abd-d$pscore)/(1-d$pscore))/ nrow(d[which(d$abd==1),])
```

This gives us an estimate of -1.0022326.

iv) (**Optional**) Use the bootstrap to estimate the SE of your IPW estimate, as well as a 95% bootstrap confidence interval. Be sure to account for uncertainty in the estimation of propensity scores as well. Do your results accord with your previous findings? Why or why not?

```
#Bootstrapped SEs and CIs for IPW estimators:
sims = 2500
att=c()
for(i in 1:sims){
  d.star = d[sample(1:nrow(d),replace=T),]
  ps.model = glm(abd ~ C.ach + C.akw + C.ata + C.oro
                 + C.paj + age, data=d, family = binomial(link=logit))
  d$pscore = predict(ps.model, type="response")
  #ATT:
  att[i] = sum(d.star$educ * (d.star$abd-d.star$pscore)/(1-d.star$pscore)) /
            nrow(d.star[which(d.star$abd==1),])
}
```

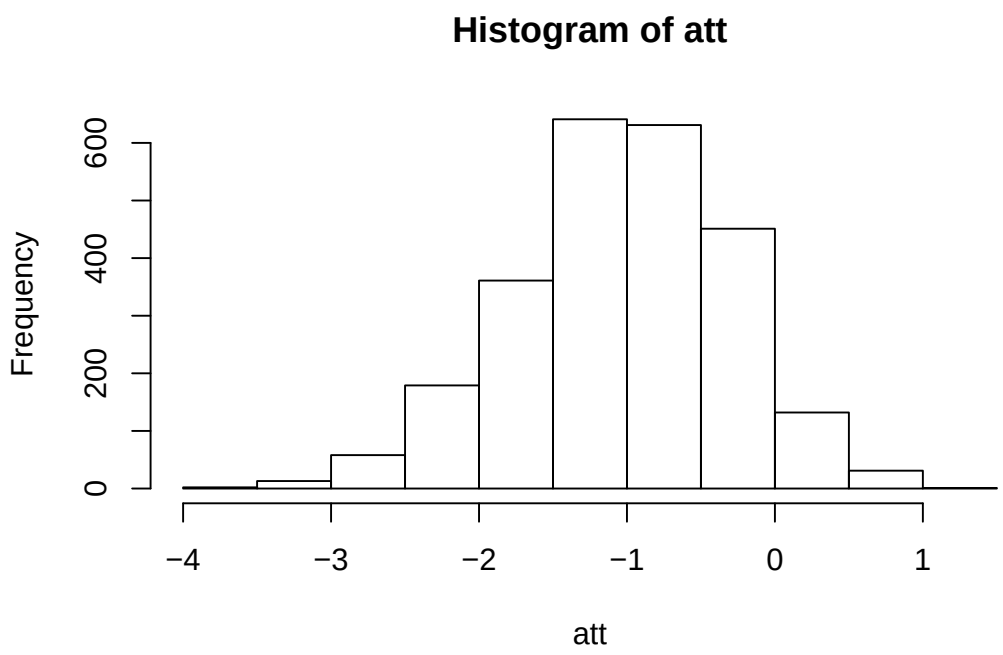
```
mean(att)
```

```
## [1] -1.025948
```

```
sd(att)
```

```
## [1] 0.7286379
```

```
hist(att)
```




```
unname(quantile(att,0.025));unname(quantile(att,0.975))
```

```
## [1] -2.544287
## [1] 0.3362582
```

You'll observe that the confidence intervals now overlap 0, suggesting that the result is no longer statistically significant at the 5% level. This is probably due to the fact that we're now estimating the uncertainty in the propensity scores – in the matching exercise earlier we assumed the p-scores were estimated without error, that is, we took the predicted probability as free from uncertainty. Now that we fold in that additional source of uncertainty, our results are less sanguine. Another potential source of increased uncertainty may be the structure of the estimator itself. Since the weighting estimator is a ratio estimator (i.e. the denominator is also estimated) it tends to be unstable when some propensity scores are close to either zero or one. To check whether this is driving our result, we can examine the distribution of bootstrap estimates. Given that

the distribution looks normal around $\hat{\tau}$, it suggests this is probably not the primary source of increased uncertainty.

Optional Problems

Problem A

The *curse of dimensionality* makes it difficult to work in a situation where there are many pre-treatment covariates to condition on. Suppose we have covariates X_k for $k = 1, \ldots, P$, where P is the number of pre-treatment covariates (i.e., the dimensionality of the covariate space). Then x_i is a vector of covariate values for observation i, $x_i = [x_{i1}, \ldots, x_{iP}]^T$.

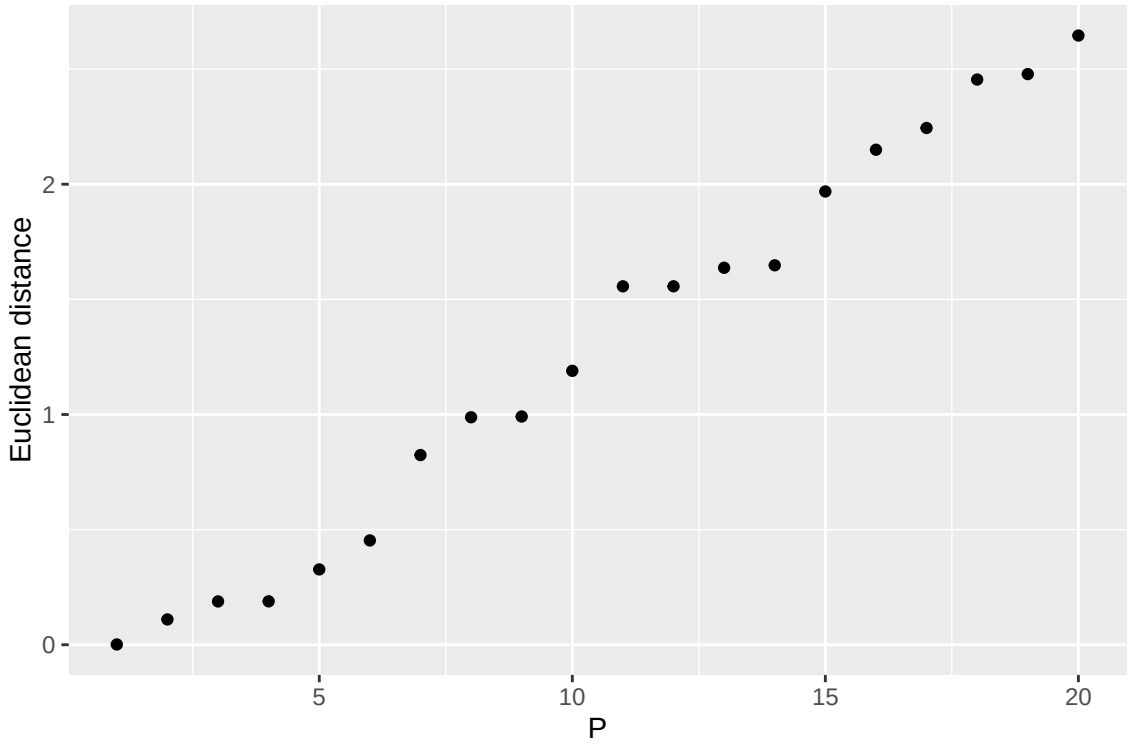
(a) Write an expression that gives the Euclidean distance between observations i and j in terms of their covariates x_i and x_j, respectively.

If $x_i = [x_{i1}, x_{i2}, \ldots, x_{iP}]$ and $x_j = [x_{j1}, x_{j2}, \ldots, x_{jP}]$, the Euclidean distance is defined as $\sqrt{\sum_{k=1}^{P}(x_{ki} - x_{kj})^2}$

(b) Create a dataset X with 500 observations and 20 covariates, where each covariate should be independently drawn from $\mathcal{N}(0, 1)$. Now, consider a new observation with covariates $x^\star$ all equal to zero. Find the new observation's nearest neighbor in the dataset you created, using your expression for Euclidean distance and only the first covariate. That is, find the point i in the dataset whose first covariate x_{1i} is closest to $x_1^\star$. Record the Euclidean distance between x_{1i} and $x_1^\star$. Repeat this process, each time adding an additional covariate: next using two covariates, then three, and so on through all 20. Use your results to plot the Euclidean distance to the new observation's nearest neighbor as a function of dimensionality.

```
library(ggplot2)

## Warning:  package 'ggplot2' was built under R version 3.2.3

# Create the data, n x P, using draws from the standard normal
n = 500
p = 20
set.seed(2139)
x = matrix(rnorm(n*p, 0, 1), ncol = p)
# A readable way to approach this is looping over rows i for
# 1:k of the p columns, calculating the Euclidean distance
# to each unit in turn.
d = c()
for (k in 1:p){
```

```
    row.dist = c()
    for (i in 1:n){
        # Euclidean distance is the sqrt of the sum of squared distances.
        # Compute it between our x* with covariates 0 and each of the n observations.
        row.dist[i] = sqrt(sum((x[i, 1:k] - 0)^2))
    }
    # Find the nearest neighbor by Euclidean distance among the n units.
    d[k] = min(row.dist)
}
# Plot distance as a function of dimensionality
ggplot(data.frame('P'=1:20, 'distance'=d), aes(x=P, y=d)) +
  geom_point() + ylab("Euclidean distance")
```



(c) What do your results demonstrate about matching?

The distance to the nearest neighbor increases linearly with the number of dimensions. (It can be proven that with variables of equal variance, as long as x_i are chosen i.i.d., the Euclidean distance increases linearly with dimensionality regardless of the covariance of the variables. The proximity of matches, or their comparability, will worsen as more continuous covariates are matched on.

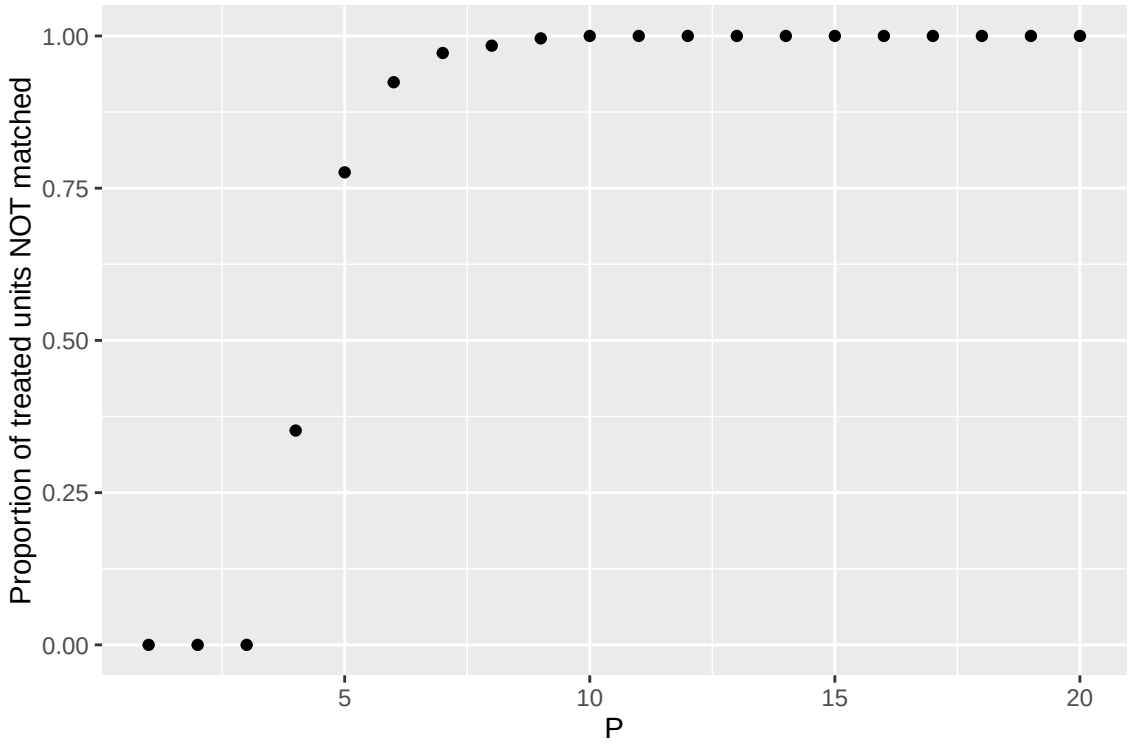
(d) We'll now examine how the curse of dimensionality affects matching on discrete covariates. Create another dataset with 500 observations, in which a treatment D_i is allocated so that

the first 250 units in the data are treated and the rest are control. Draw each of the 20 covariates independently from the multinomial distribution with 4 categories, with probabilities $p = [0.1, 0.2, 0.3, 0.4]$ for the treatment group and $p = [0.25, 0.25, 0.25, 0.25]$ for the control group. Now, match each treated unit to an untreated unit with exactly the same x_{i1}, with replacement. Note the proportion of treated units that can be matched, and repeat as in (b) with increasing dimensionality up to 20. Plot the proportion of unmatched treated units as a function of dimensionality. What do you conclude? How would your plot change with increasing units in your data?

In the sample drawn for the solutions, we can find exact matches for all of our treated units with up to three discrete covariates. In higher dimensions not all treated units are matched. This is significant because we can drop untreated units and still identify the ATT, as seen in Problem 1 **??**, but cannot after dropping any treated unit. We can also see that the difficulty of finding binary matches increases exponentially. (The volume of the covariate space increases exponentially in P.) With more units, we could afford to match exactly on more discrete covariates, but this relationship would hold.

```
library(ggplot2)
n = 500
p = 20
# Create a treatment vector that assigns the first 250 units to treatment;
# it'll be easier to keep this separate from the covariate data
d = 1:n %in% 1:250
# Draw n x 20 covariates from the Multinomial
set.seed(2139)
x = sapply(1:20, function(k) {
  # Draw x_k for treated units, and then control units
  bins = rmultinom(250, 1, prob=c(.1, .2, .3, .4))
  bins = cbind(bins, rmultinom(250, 1, prob=rep(.25, 4)))
  # The result is a 4 x 500 matrix in which the row number gives
  # the value of our categorical variable
  apply(bins, 2, function(i) which(i == 1))
})
# Again loop over rows i for 1:k of the p columns, this time
# checking for exact matches
matched = c()
for (k in 1:p){
    matches = sapply((1:n)[d], function(i) {
    # Check for equality between covariates for treated unit i and each untreated
    # unit; the == operator is vectorized column-wise and we want to
    # compare row-wise, so transpose the array of covariates for the
```

```
    # untreated. We get back an array of the same dimensions with T/F for
    # each covariate. For a control unit that matches on all covariates, its
    # column will contain k TRUEs, and the sum of the column will be k.
    # To count the exact matches, sum over the column sums == k.
        sum(colSums(x[i, 1:k] == t(x[!d, 1:k])) == k)})
    # matches is a 250-vector with the count of matches for each treated i; we want the
    # proportion of treated units matched, which is the ratio of non-zero counts
    # in the vector to treated units
    matched[k] = sum(matches > 0, na.rm=T) / length(matches)
}

unmatched<-1-matched

# Plot distance as a function of dimensionality
ggplot(data.frame('P'=1:20, 'unmatch rate'=unmatched), aes(x=P, y=unmatch.rate)) +
  geom_point()+ylab("Proportion of treated units NOT matched")
```



(e) Now we'll examine the curse of dimensionality as it relates to distance metric-based matching. Using the same dataset you created in (d), match each treated unit to an untreated unit using nearest neighbor matching with the Mahalanobis distance (see Part II, Slide 3 in the Observational Studies lecture slides for the definition). Again start by finding matches and calculating the distance between them based on a single covariate, then two, then three, until

you reach $P = 20$. Plot the mean Mahalanobis distance between nearest neighbor matches across all treated units as a function of P. How do your results differ from those in part (d)?

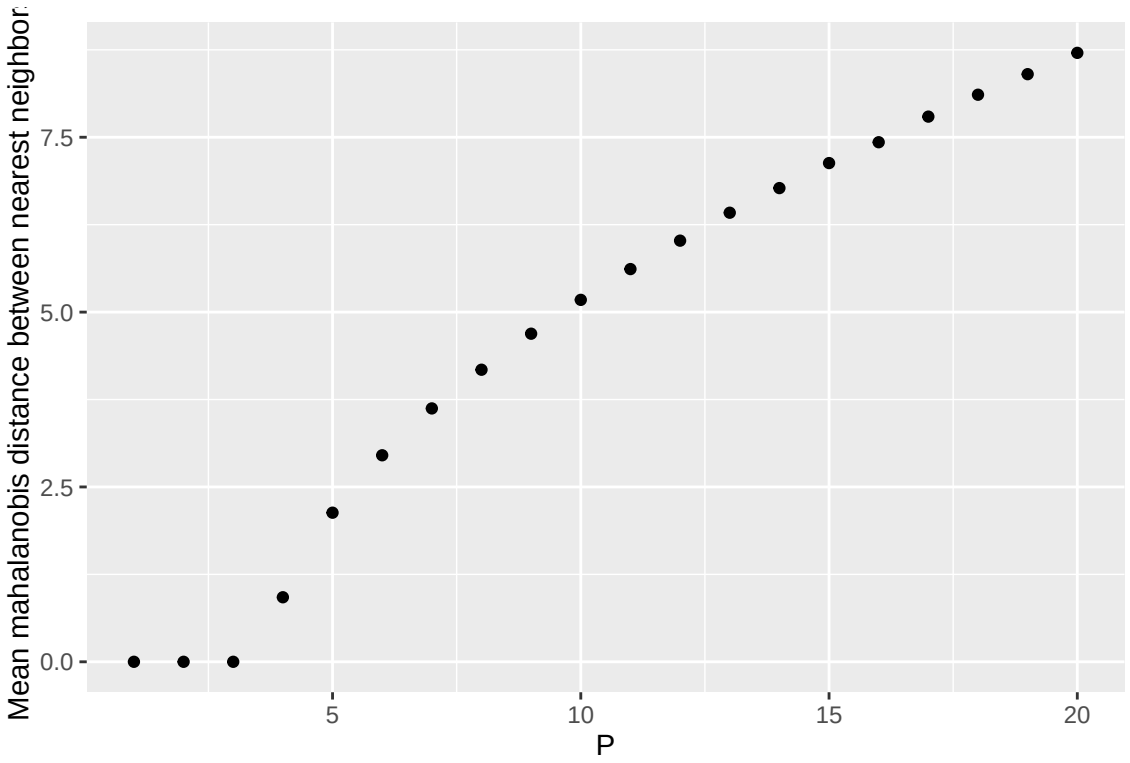
(Note: Please write your own code to calculate the Mahalanobis distance between observations — do not use the canned function `mahalanobis()`. Also notice that our pre-treatment covariates are categorical, meaning that we will need to first transform them into appropriate sets of dummy variables before calculating distances.)

Here, the dimensionality problem seems substantially reduced compared to the case of exact matching. As expected based on our results from (d) we see that for the first 3 covariates we are able to find exact matches (the distance between treated and control units is zero). As we increase the number of covariates the average mahlanobis distance between treated and control nearest neighbors increases. However, compared to (d) we are now using a single dimension to summarize the covariate space and in essence have "solved" the problem of distance between nearest neighbors exponentially increasing as P increases. This is same result as in (a), where we used the Euclidian distance metric. Of course, this achievement may be illusory — if the confounding cannot be captured well in terms of the summary distance measure of Mahalanobis distance, the matching procedure may not lead to improvement in terms of bias.

```
### first we need to make our categorical dataset into binary variables ###
# since it's not meaningful to measure distance between categorical
# variables other than 0/1 we need to create a new dataset with dummy
# variables for each categorical variable value
# now instead of a dataset of 500 x 20 observations
# we will have a dataset of 500 x 60 observations
# since we will create 3 new binary variables for each old categorical
# variable i.e. x_1_1 = 0/1 if x_1 == 1, x_1_2 = 0/1 if x_1 == 2,
# and x_1_3 = 0/1 if x_1 == 3  (the 4th binary variable is removed
# because it's a linear combination of the others)

# make our matrix into a dataframe with factor variables
x<-as.data.frame(apply(x,2,as.factor))

# convert factor variables into 3 dummy variables each
# model.matrix automatically removes the 4th, collinear, dummy
xnew<-model.matrix(~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8
                   + V9 + V10 + V11 + V12 + V13 + V14 + V15
                   + V16 + V17 + V18 + V19 + V20,data=x)[,-1]
                   # delete default intercept column
# now we have a 500 x 60 dataset
```

```r
# now that we have the dataset setup
# split datasets into treatment and control
treated<-xnew[1:250,]
control<-xnew[251:500,]

######## calculate Mahalanobis distance ########
# matrix to store calculated minimum MDs (for each i,k combination)
MDmini<-matrix(NA,250,p)

for(k in 1:p){
  for(i in 1:250){
    # create treament matrix -- copy the i'th row 250 times
    # 250 x (k*3) matrix of repeated ith treatment observation
    tmat<-matrix(as.numeric(treated[i,1:(k*3)]),250,k*3,byrow=T)

    # define control matrix (all 250 control rows)
    # 250 x (k*3) matrix of all control observations
    cmat<-as.matrix(control[,1:(k*3)])

    # calculate the distance between X_t and X_c
    # this will be a 250 x (k*3) matrix
    XtXc<-as.matrix(tmat-cmat)

    # calculate the variance-covariance matrix of the entire dataset,
    # a (k*3) x (k*3) varcov matrix
    Xsigma<-var(xnew[,1:(k*3)])

    # calculate the MD for the ith treated row to ALL 250 control rows
    # the diagonal of the 250 x 250 matrix = distance measures we want
    # (the distance between treated row i and all control observations)
    MDiallj<-sqrt(diag(XtXc %*% solve(Xsigma) %*% t(XtXc)))

    # record the minimum distance
    # between treated observation i and all control observations
    MDmini[i,k]<-min(MDiallj)
  }
}

# calculate mean minimum MD across all treated units (for each value of k)
```

```r
meanMDmin<-colMeans(MDmini)

# Plot distance as a function of dimensionality
ggplot(data.frame('P'=1:20, 'MD'=meanMDmin), aes(x=P, y=meanMDmin)) +
  geom_point()+ylab("Mean mahalanobis distance between nearest neighbors")
```



Problem B

In this problem we revisit Blattman and Annan's data on child soldiers in Uganda (`child_soldiering.csv`), which we used in Problem 2.

(a) Estimate the ATE of abduction on years of education with OLS. Fit the model

$$Y_i = \hat{\alpha} + \hat{\beta} D_i + \hat{\gamma} \mathrm{X} + u_i$$

Where $\hat{\alpha}$ is an intercept, $\hat{\beta}$ is the OLS estimator for the ATE, and $\hat{\gamma}$ is a length-four vector of coefficients for X, which is an $N \times 4$ matrix of possible observed confounders: age, father's education, mother's education, and residency in Acholibur (`C.ach`); residuals appear as u_i. If we want to interpret the ATE estimate produced from this regression model as the causal effect of abduction on years of education what is our identification assumption? What additional assumptions are required for the OLS estimator $\hat{\beta}$ to be unbiased and consistent for the ATE?

The identifying assumption is the independence of treatment and potential outcomes conditional on the observed covariates, $Y_{1i}, Y_{0i} \perp\!\!\!\perp D_i | X_i$ (i.e. conditional ignorability). We also need the common support assumption, $0 < Pr(D_i = 1 | X_i = x) < 1$. For the OLS estimator $\hat{\beta}$ to be unbiased and consistent for the ATE, further assumptions of constant treatment effect and linearity must hold, but note that these are not *identifying* assumptions. The identification result is unrelated to our choice of estimator.

```
library(stargazer)
library(lmtest)
library(sandwich)

d = read.csv('child_soldiering.csv')

# Use the regression estimator
mod1 = lm(educ ~ abd + age + fthr.ed + mthr.ed + C.ach, data=d)
mod1HC2 = list(coeftest(mod1, vcov = vcovHC(mod1, type = "HC2"))[,2])

stargazer(mod1, se = mod1HC2, title = 'OLS Results',
          style='apsr', keep.stat=c('n'), notes=c("HC2 Robust SEs"))
```

Table 7: OLS Results

	educ
abd	-0.554^{**}
	(0.218)
age	0.046^{**}
	(0.021)
fthr.ed	0.159^{***}
	(0.031)
mthr.ed	0.053
	(0.041)
C.ach	-0.716^{**}
	(0.290)
Constant	5.464^{***}
	(0.454)
N	741

*p $<$ .1; **p $<$ .05; ***p $<$ .01
HC2 Robust SEs

(b) Thinking about the context of the experiment (see the paper for more details), come up with a possible binary confounder that predicts both education and abduction. Describe your

binary confounder, in words, and then draw a DAG to represent the relationships between abduction (D), education (Y) and your potential confounder (U).

One potential unobserved binary confounder in this setup might be the presence of NGO's in the village. If there is a large NGO presence in an area rebels might be deterred from conducting raids in that area (i.e. a negative relationship between U and D). The presence of NGO's might also improve the availability of or access to education in the area, increasing the probability that students attend school (i.e. a positive relationship between U and Y).

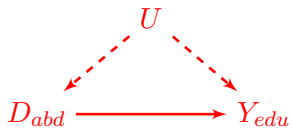


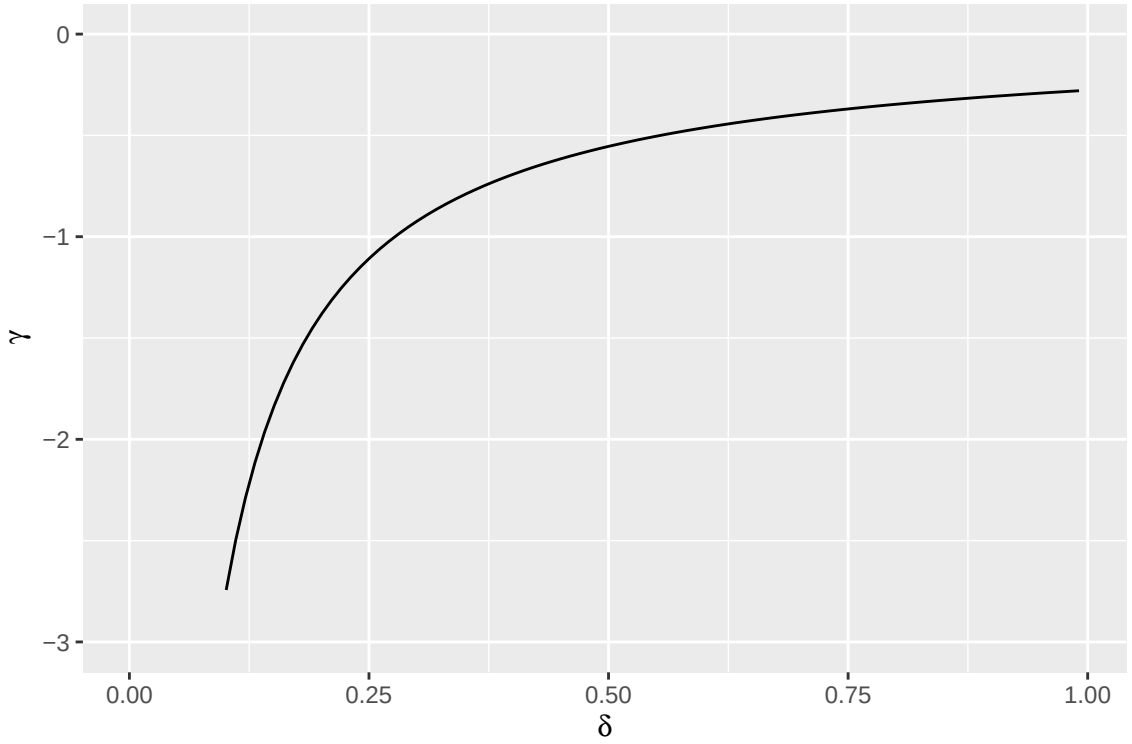
Figure 3: DAG with confounding variable

(c) Consider your confounding variable, how plausible is the assumption that D_i and U_i do not interact, in other words that the average effect of U_i on Y_i is constant between treatment groups? For now, let's say this assumption is plausible and let's examine this binary confounder that predicts both education and abduction.

Create a contour plot of the bias that would reduce the magnitude of the ATE by half, where the x-axis is δ (the difference in average U_i between treatment conditions or imbalance in U) and the y-axis is γ (the effect of U_i on Y_i). Refer to slide 13 of the Observational Studies Part III: Nonparametric Bounds and Sensitivity Analysis lecture for more precise definitions of these sensitivity parameters (δ and γ).

In our example it is unlikely that this assumption is valid because NGO presence likely has a much larger, positive, effect for individuals who are not abducted (control) compared to those who are abducted (treated). But for now we make this assumption and plot the halved ATE.

```
library(ggplot2)
# What is half the magnitude of the treatment coefficient?
b.half = mod1$coefficients[2] / 2
# The range of delta is 0,1
delta = seq(.001, 1, by = .01)
# Solve for gamma
g = b.half / delta

# Make plot
ggplot(data.frame(delta=delta, g=g), aes(x=delta, y=g)) + geom_path() +
  xlab(expression(delta)) + ylab(expression(gamma)) + ylim(-3,0)
```



(d) Now we can see the values of δ and γ that would reduce our ATE by half. However, in order to get a sense of whether it is plausible, and even likely, that our unobserved confounder could create this amount of bias we need to compare it to something. Compare the hypothetical degree of confounding based on the values of our sensitivity parameters with the observed degree of confounding based on observed covariates.

Add the covariate residency in Acholibur (`C.ach`) to the plot as a benchmark. Assume that the covariates (`age`) and (`fthr.ed`) have a negative relationship (but the same magnitude) with our outcome, education. Add these observed covariates to the plot as well where the x-axis value is the change in propensity scores when moving from the first to third quartiles of these two variables.

Label all the points on your plot. Discuss how predictive the unobserved confounder would need to be in relation to the benchmarks to reduce the ATE by half.

The coefficient on `C.ach` in our OLS model can be plotted along γ in the scale of Y_i. When we make the coefficients on `age` and `fthr.ed` negative we can do the same. Estimating the expected difference in propensity scores associated with `C.ach` and moving from the first to the third quartile for `age` and `fthr.ed` will give us their corresponding δ's. In the solutions code, we estimate propensity scores as the predicted values from a logit regression of treatment on observables, just as when matching or weighting on the propensity score, but we do so twice: from model matrices in which `C.ach` is set to 1 and then to 0 (`age` and `fthr.ed` set to the first quartile value and then the third). The expected difference in the resulting propensity score estimates is the estimated change in the probability of treatment,

conditional on observables, associated with the possible binary confounder.

Comparing the location of `C.ach`, `age` and `fthr.ed` on the plot to the combinations of δ and γ for U_i that would halve the ATE, we can see that an unobserved binary confounder U_i would need to be about 50 percent more predictive of abduction and education than `C.ach` to reduce the ATE by half. If unobserved confounding is only as strong as these observed covariates then our results are robust in the sense that our ATE may be reduced, but by less than half and is still negative.

```
# Add covariates as benchmarks to the plot

# Estimate delta for covariates: fit a propensity score model
mod.p = glm(abd ~ age + fthr.ed + mthr.ed + C.ach, data=d,
            family=binomial(link='logit'))

# Extract the model matrix
x.mod.p = model.matrix(mod.p)

# Create a copy of the model matrix in which C.ach is only 1
x.mod.p.ach1 = data.frame(x.mod.p[, 1:4], C.ach=1)

# And another where C.ach is only 0
x.mod.p.ach0 = data.frame(x.mod.p[, 1:4], C.ach=0)

# Get predicted probabilities
pscore.ach1 = predict(mod.p, newdata=x.mod.p.ach1, type="response")
pscore.ach0 = predict(mod.p, newdata=x.mod.p.ach0, type="response")

# The difference in means is the change in propensity from
# having C.ach=1 instead of C.ach=0, conditional on
# other covariates
delta.ach = mean(pscore.ach1) - mean(pscore.ach0)
ach.point = data.frame(delta=delta.ach, g=mod1$coefficients[6])

# do the same process for Age
summary(d$age)# Q1:age = 17 and Q3:age=25

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.00   17.00   20.00   20.91   25.00   30.00

# Create a copy of the model matrix in which age = 17
```

```r
x.mod.p.youth1 = data.frame(x.mod.p[, c(1,3:5)], age=17)

# And another where age = 25
x.mod.p.youth0 = data.frame(x.mod.p[, c(1,3:5)], age=25)

# Get predicted probabilities
pscore.youth1 = predict(mod.p, newdata=x.mod.p.youth1, type="response")
pscore.youth0 = predict(mod.p, newdata=x.mod.p.youth0, type="response")

# diff in means (change in propensity scores) and corresponding gamma
delta.youth = abs(mean(pscore.youth1) - mean(pscore.youth0))
youth.point = data.frame(delta=delta.youth, g=mod1$coefficients[3])


# do the same process for father's education (1st to 3rd quartile)
summary(d$fthr.ed) #Q1 = 4, Q3 = 10

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.000   4.000   5.879  10.000  15.000

# Create a copy of the model matrix in which fthr.edu=4
x.mod.p.fedu1 = data.frame(x.mod.p[, c(1:2,4:5)], fthr.ed=4)

# And another where fthr.edu=10
x.mod.p.fedu0 = data.frame(x.mod.p[, c(1:2,4:5)], fthr.ed=10)

# Get predicted probabilities
pscore.fedu1 = predict(mod.p, newdata=x.mod.p.fedu1, type="response")
pscore.fedu0 = predict(mod.p, newdata=x.mod.p.fedu0, type="response")

# diff in means (change in propensity scores) and corresponding gamma
delta.fedu = abs(mean(pscore.fedu1) - mean(pscore.fedu0))
fedu.point = data.frame(delta=delta.fedu, g=mod1$coefficients[4])

# Make plot
ggplot(data.frame(delta=delta, g=g), aes(x=delta, y=g)) + geom_path() +
  xlab(expression(delta)) + ylab(expression(gamma)) + ylim(-3,0) +
  geom_point(data=ach.point, aes(x=delta, y=g, label='Acholibur')) +
  geom_text(data=ach.point, aes(x=delta, y=g, label='Acholibur'),
            vjust=-.2, hjust=-.1) +
```

```
  geom_point(data=youth.point, aes(x=delta, y=(-1)*g, label='Age')) +
  geom_text(data=youth.point, aes(x=delta, y=(-1)*g, label='Age'),
            vjust=-.2, hjust=-.1) +
  geom_point(data=fedu.point, aes(x=delta, y=(-1)*g, label='Fthr.ed')) +
  geom_text(data=fedu.point, aes(x=delta, y=(-1)*g, label='Fthr.ed'),
            vjust=-.2, hjust=-.1)
```