R Notebook

Set up

```
libs <- c('MatchIt', "ggplot2")
needed <- setdiff(libs, .packages(all = TRUE))
if (length(needed) > 0) {install.packages(needed); print("Yay!")}
for (lib in libs) {library(lib, character.only = TRUE)}
```

Data Generation

```
set.seed(123)

N = 5000
a = rnorm(N)
b = rnorm(N)

# True Treatment effect = 2 (i.e. 102-100, the other parts just add noise of
# differing variance)
# note: y1 has twice the slope of y0 on both variables. But we are only interested in
# the AVERAGE binary treatment effect over all levels of a and b, rather than
# prediction accuracy at particular levels (at which treatment effect may vary
# greatly from its avg as it does here). The ATE is equivalent to the difference of
# in y-intercept  bc our data (both variables a and b) are centered around x=0
# because they are drawn from ~N(0,1) dist.
y1 = 102 + 6*a + 4*b + rnorm(N)
y0 = 100 + 3*a + 2*b + rnorm(N)

u = (a+b)/2 # why divide by 2?
p_d_given_a_b = plogis(u)
d = rbinom(rep(1,N), 1, p_d_given_a_b)

y = d * y1 + (1-d) * y0

data = data.frame(d, y, a, b, u)

sum(d)/length(d)
```

```
## [1] 0.5014
```

```
# 50.14% in treatment group
```

naive estimate (of ATE?), no cofounds controlled for

```
# the coefficient for D represents the niave treatment effect

model = lm(y ~ d, data)
summary(model)
```

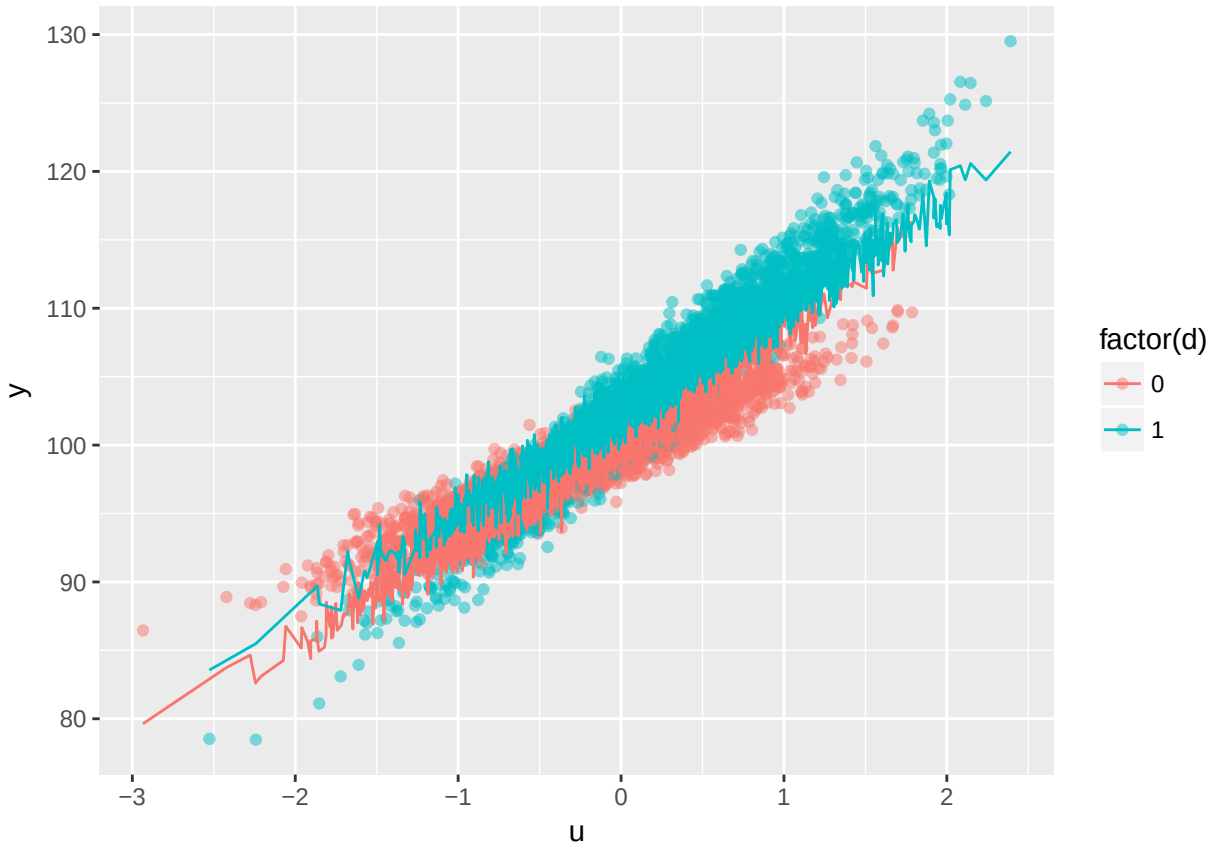
```
##
## Call:
## lm(formula = y ~ d, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.933  -3.228  -0.017   3.216  25.108
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  98.7519     0.1102  896.25   <2e-16 ***
## d             5.6563     0.1556   36.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.501 on 4998 degrees of freedom
## Multiple R-squared:  0.2091, Adjusted R-squared:  0.2089
## F-statistic:  1321 on 1 and 4998 DF,  p-value: < 2.2e-16
```

regression with conditioning estimate of ATE

```
model <- lm(y ~ d+a+b, data)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ d + a + b, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2709 -1.3244 -0.0281  1.3373  8.0759
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 100.62030    0.04073 2470.48   <2e-16 ***
## d             1.96058    0.05933   33.04   <2e-16 ***
## a             4.56318    0.02875  158.69   <2e-16 ***
## b             3.05566    0.02881  106.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.964 on 4996 degrees of freedom
## Multiple R-squared:  0.8992, Adjusted R-squared:  0.8992
## F-statistic: 1.486e+04 on 3 and 4996 DF,  p-value: < 2.2e-16
```

```
# d coef = 1.96058
# this is close to accurate!

# look at data by treatent status
ggplot(data) +
  geom_point(aes(x=u, y=y, col= factor(d)), alpha = .5) +
  geom_line(aes(x = u, y=predict(model), group=d, col= factor(d))) #+
```




```
  #geom_line(aes(x = u, y=predict(model2), group=d), col= "orange")
```

Manual Pronensity score

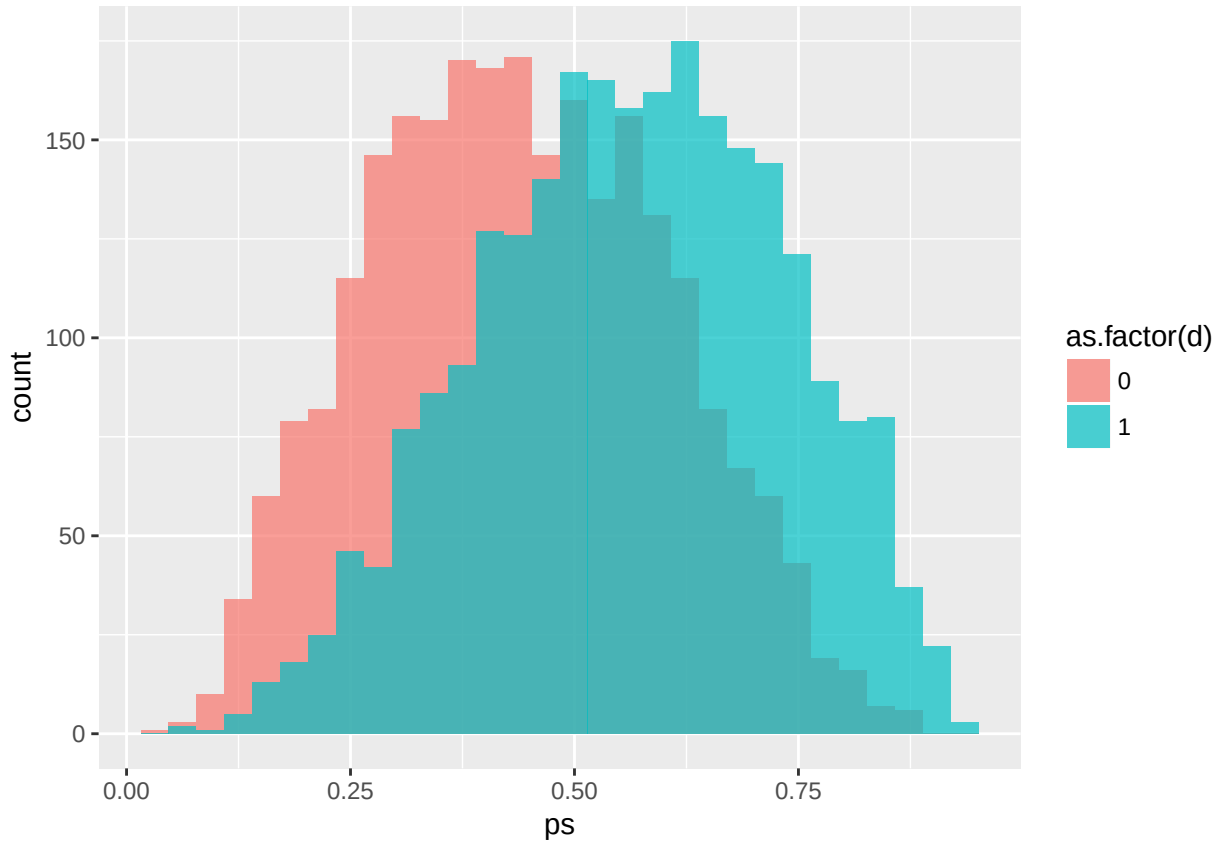
Create propensity scores

```
fit <-  glm(d~a+b, family = binomial(link = "logit"), data)
propensity <- predict(fit, type = "response") # P(D | Zs)
data$ps <- propensity
```

Check Shared Support on propensity scores between treatment and control

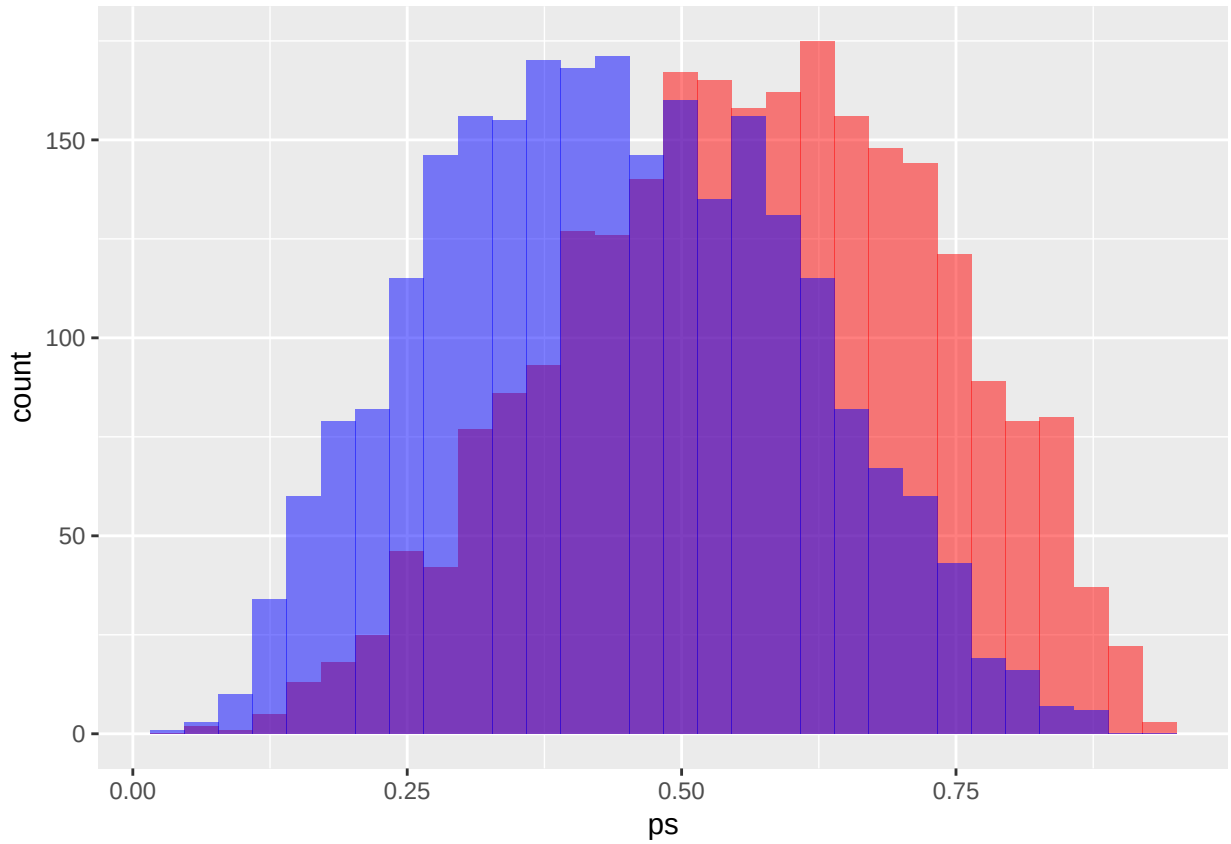
```
# overlapping histogram
ggplot() +
  geom_histogram(data = data, mapping = aes(x= ps, fill = as.factor(d)), alpha = .7, position= "identity
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# identical result
ggplot() +
  geom_histogram(data = data[data$d == 1,], mapping = aes(x= ps), alpha = .5, fill = "red") +
  geom_histogram(data = data[data$d == 0,], mapping = aes(x= ps), alpha = .5, fill = "blue")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# percent of data unsupported
 shared_suport <- sort(c(range(data[data$d == 1, "ps"]), range(data[data$d == 0, "ps"])))[2:3]
 unsupported_rows <- (data$ps < shared_suport[1]) |  (data$ps > shared_suport[2])
# percent of data unsupported
 (sum(unsupported_rows)/nrow(data)) * 100
```

```
## [1] 0.82
```

```
# 0.82% .. under 1 percent of data set is unsupported
```

Match controls to treatment

```
# Q: why can't we just match without propensity scores using knn in the Z vars?

###
match <- function(data, D_col, PS_col) {
  treated <- data[data[D_col] == 1,]
  control <- data[data[D_col] == 0,]

  nearest <- function(cell, match_with = c("treated", "control")){
    assignment <- match.arg(match_with)
    df <- list(treated = treated, control = control)[[assignment]]
    nearest_loc <- which.min(abs(cell - df[[PS_col]]))
    nearest_loc
  }
```

```
  # match controls to treateds
  treated$matched_controls_idx <- apply(treated[PS_col], 1, function(cell){nearest(cell, "control")})

  treated$matched_controls_y <- control$y[treated$matched_controls_idx]

  # match treateds to controls
  control$matched_treateds_idx <- apply(control[PS_col], 1, function(cell){nearest(cell, "treated")})

  control$matched_treateds_y <- treated$y[control$matched_treateds_idx]

  list(treated = treated, control = control)
}

#####

get_estimates <- function(matched_out) {
  stopifnot("list" %in% class(matched_out))
  stopifnot(c("treated", "control") %in% names(matched_out))
  #stopifnot(c("d", "y"))

  # ATT
  y1_d1 <- mean(matched_out$treated$y)
  y0_d1_hat <- mean(matched_out$treated$matched_controls_y)
  ATT <- y1_d1 - y0_d1_hat

  #ATC
  y1_d0_hat <- mean(matched_out$control$matched_treateds_y)
  y0_d0 <- mean(matched_out$control$y)
  ATC <- y1_d0_hat - y0_d0

  #ATE
  all <- rbind(matched_out$treated[,1:6], matched_out$control[,1:6])
  all$y1 <- all$y
  all$y0 <- all$y
  all[all$d ==1, "y0"] <- matched_out$treated$matched_controls_y
  all[all$d ==0, "y1"] <- matched_out$control$matched_treateds_y
  ATE <- mean(all$y1 - all$y0)

  return(list(ATT = ATT, ATC = ATC, ATE = ATE))
}
#####

test <- match(data, D_col = "d", PS_col = "ps")

## Check balance
# for "a" var
mean(test$treated$a) - mean(test$control$a[test$treated$matched_controls_idx])
```

```
## [1] -0.0154599
```

```
# -0.0154599 ... fairly close to zero


## get ATT, ATC, and ATE estimates
```

```r
y1_d1 <- mean(test$treated$y)
y0_d1_hat <- mean(test$treated$matched_controls_y)
ATT <- y1_d1 - y0_d1_hat
# 3.166293... i.e. correct treatment

y1_d0_hat <- mean(test$control$matched_treateds_y)
y0_d0 <- mean(test$control$y)
ATC <- y1_d0_hat - y0_d0
# 0.7619367


##!!!!### is this a weighted avg?.. NO itseems
ATE <- (ATT + ATC)/2
# 1.964115 ... quite close to 2


all <- rbind(test$treated[,1:6], test$control[,1:6])

all$y1 <- all$y
all$y0 <- all$y
all[all$d ==1, "y0"] <- test$treated$matched_controls_y
all[all$d ==0, "y1"] <- test$control$matched_treateds_y

mean(all$y1 - all$y0)
```

```
## [1] 1.967481
```

```r
# 1.967481... even slightly closer!
# The difference from ATE calculated from ATT and ATC above is likely due to the this
# being ineffect a weighted avg of ATT and ATC based on the slight imbalance in the
# size of the treatment and control groups
```

Calculate ATT, ATC (, ATE) with Matchit package

Match control units to treatment units to approximate hypothetical Y_0 of treated units and visa versa

```r
  # att model
 result <- matchit(d ~ a + b, data, method = "nearest", distance = "logit", replace=TRUE)
  matched_data_att = match.data(result)
  att_model = lm(y ~ d, data=matched_data_att, weights=matched_data_att$weights)


  # atc model
  data$d <- (data$d + 1) %% 2  # flip the assignment to match treatement to control
  result <- matchit(d ~ a + b, data, method = "nearest", distance = "logit", replace=TRUE)
  matched_data_atc = match.data(result)
  data$d <- (data$d + 1) %% 2  # flip it back for further use
  matched_data_atc$d <- (matched_data_atc$d + 1) %% 2  # revert to get correct treatment effect estimat
  atc_model = lm(y ~ d, data=matched_data_atc, weights=matched_data_atc$weights)

  #atc_estimates[[i]]<-atc_model$coefficients[[2]]
```

```
  #att_estimates[[i]]<-att_model$coefficients[[2]]
  #att_errs[[i]]<-coef(summary(att_model))[,2][[2]]
  #atc_errs[[i]]<-coef(summary(atc_model))[,2][[2]]
```

check balance of Zs between groups

```
summary(result)
```

```
##
## Call:
## matchit(formula = d ~ a + b, data = data, method = "nearest",
##     distance = "logit", replace = TRUE)
##
## Summary of balance for all data:
##          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean
## distance        0.5608        0.4367     0.1676    0.1241  0.1294   0.1243
## a              -0.2273        0.2249     0.9781   -0.4522  0.4484   0.4509
## b              -0.2720        0.2622     0.9761   -0.5342  0.5325   0.5328
##          eQQ Max
## distance  0.1398
## a         0.6044
## b         0.9711
##
##
## Summary of balance for matched data:
##          Means Treated Means Control SD Control Mean Diff eQQ Med eQQ Mean
## distance        0.5608        0.5608     0.1627    0.0000  0.0582   0.0549
## a              -0.2273       -0.2474     0.9910    0.0201  0.1906   0.1921
## b              -0.2720       -0.2543     0.9459   -0.0177  0.2370   0.2435
##          eQQ Max
## distance  0.0703
## a         0.4154
## b         0.4433
##
## Percent Balance Improvement:
##          Mean Diff. eQQ Med eQQ Mean eQQ Max
## distance    99.9870 54.9927  55.8242 49.7055
## a           95.5613 57.4904  57.4025 31.2730
## b           96.6852 55.4904  54.2970 54.3516
##
## Sample sizes:
##           Control Treated
## All          2507    2493
## Matched      1209    2493
## Unmatched    1298       0
## Discarded       0       0
```

avg both groups and take difference to get ATT

```
atc_model$coefficients[[2]]
```

```
## [1] 0.7619367
```

```
# 0.7619367... literally identical to my manual values!!

att_model$coefficients[[2]]
```

```
## [1] 3.166293
```

```
# 3.166293


# identical
matched_data_att$weighted_y = matched_data_att$weights * matched_data_att$y
disp <- aggregate(matched_data_att[,c("d","weighted_y")], list(matched_data_att$d), mean)
# y1_d1 - y0_d1
disp[2,3] - disp[1,3]
```

```
## [1] 3.166293
```

```
# ATT
# 3.166293

# identical
model <- lm(weighted_y ~ d, data = matched_data_att)
summary(model)
```

```
##
## Call:
## lm(formula = weighted_y ~ d, data = matched_data_att)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -60.27   -9.22   -2.11    4.64 1935.69
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  101.242      1.771  57.154   <2e-16 ***
## d              3.166      2.146   1.475     0.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.69 on 3679 degrees of freedom
## Multiple R-squared:  0.0005911,  Adjusted R-squared:  0.0003195
## F-statistic: 2.176 on 1 and 3679 DF,  p-value: 0.1403
```

Is PSM more robust to Treatment/Control imbalance due to Selection Bias??: YES!

```
### Data Generation
set.seed(123)

N = 5000
a = rnorm(N)
b = rnorm(N)

# True Treatment effect = 2 (i.e. 102-100, the other parts just add noise of differing variance)
```

```r
# note: y1 has twice the slope of y0 on both variables. But we are only interested in
# the AVERAGE binary treatment effect, rather than prediction accuracy at all levels
y1 = 102 + 6*a + 4*b + rnorm(N)
y0 = 100 + 3*a + 2*b + rnorm(N)

u = (a+b)/2 # why divide by 2?
p_d_given_a_b = plogis((2*u)-3) # multiply to make curve steeper and substract 3 to  # shift probabilit
d = rbinom(rep(1,N), 1, p_d_given_a_b)

y = d * y1 + (1-d) * y0

data = data.frame(d, y, a, b, u)

sum(d)/length(d)
```

```
## [1] 0.0882
```

```r
# 8.82% in treatment group


### Regression w/ conditioning estimate of ATE
model1 <- lm(y ~ d+a+b, data)
summary(model1)
```

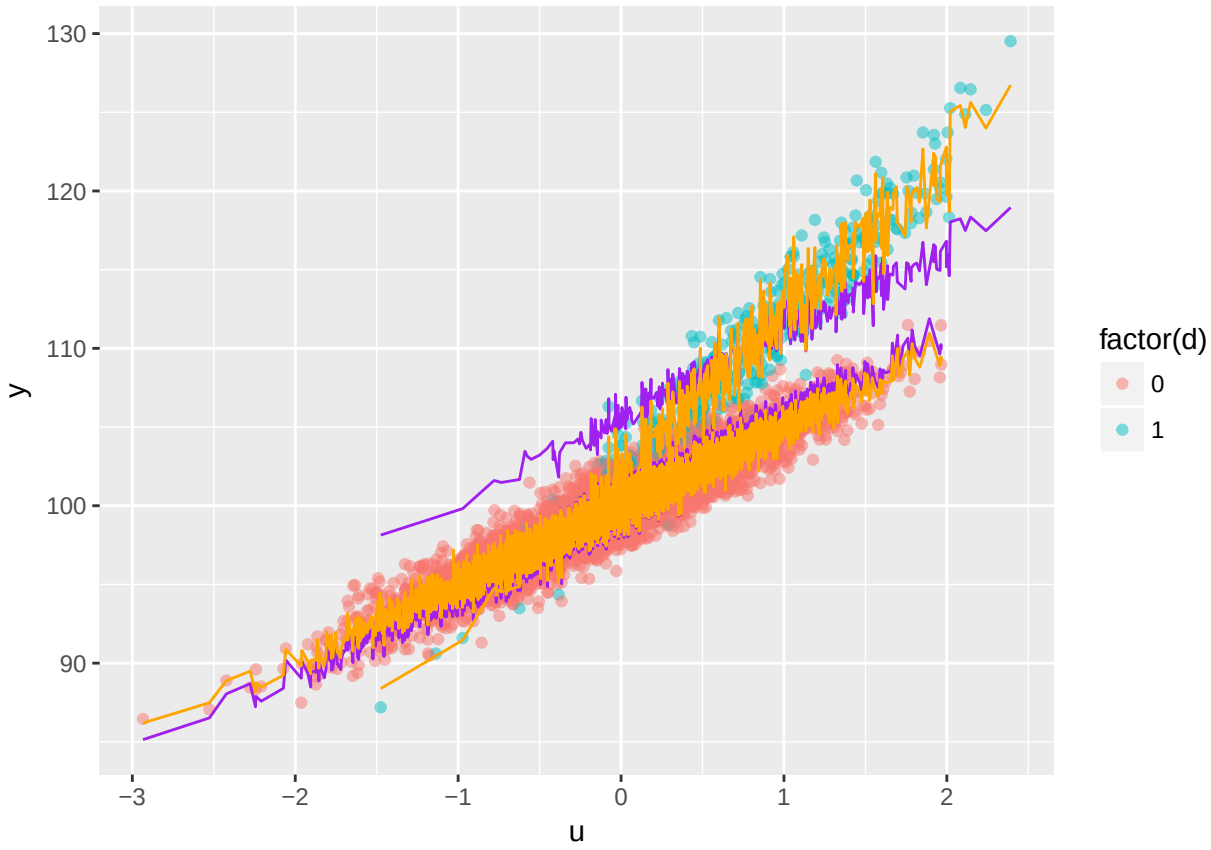
```
##
## Call:
## lm(formula = y ~ d + a + b, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9460  -0.7589   0.0039   0.7327  10.5693
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 100.00668    0.02018 4956.78   <2e-16 ***
## d             5.55926    0.07176   77.47   <2e-16 ***
## a             3.24329    0.01993  162.72   <2e-16 ***
## b             2.15986    0.01965  109.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.354 on 4996 degrees of freedom
## Multiple R-squared:  0.9218, Adjusted R-squared:  0.9217
## F-statistic: 1.962e+04 on 3 and 4996 DF,  p-value: < 2.2e-16
```

```r
# beta_d == 5.55926 ... way bigger than 2

#however if we use interactions....
# note: I only know that d:a and d:b interactions are necessary and useful but not
# a:b (which might cause over fitting...maybe), because I know the data generation
# process and I have seen the graph.. This is unrealistic
model2 <- lm(y ~ d+a+b+d:a+d:b, data)
summary(model2)
```

```
##
```

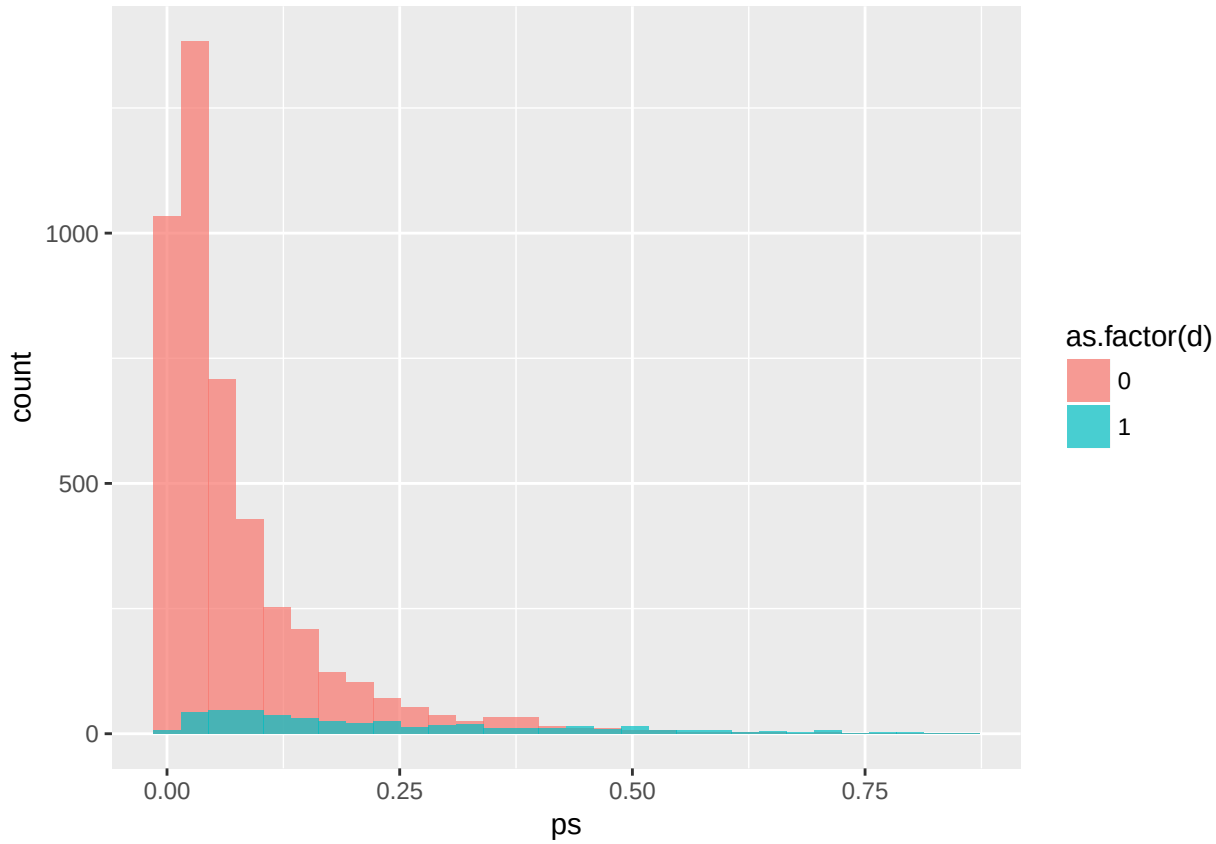
```
## Call:
## lm(formula = y ~ d + a + b + d:a + d:b, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4703 -0.6798 -0.0146  0.6605  3.7871
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 99.97601    0.01487 6721.15   <2e-16 ***
## d            2.08545    0.07588   27.48   <2e-16 ***
## a            2.99625    0.01536  195.09   <2e-16 ***
## b            2.00715    0.01513  132.63   <2e-16 ***
## d:a          2.99035    0.05263   56.82   <2e-16 ***
## d:b          1.96118    0.05214   37.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9978 on 4994 degrees of freedom
## Multiple R-squared:  0.9575, Adjusted R-squared:  0.9575
## F-statistic: 2.252e+04 on 5 and 4994 DF,  p-value: < 2.2e-16
```

```r
# beta_d == 2.08545 ... very close to True value! because model adapts to the data   # better... will t

# look at data by treatent status
ggplot(data) +
  geom_point(aes(x=u, y=y, col= factor(d)), alpha = .5) +
  geom_line(aes(x = u, y=predict(model1), group=d), col="purple") +
  geom_line(aes(x = u, y=predict(model2), group=d), col= "orange")
```




```
### PSM estimate of ATE
# make sure to read in the match() function from above first
# make propensity scores
fit <-  glm(d~a+b, family = binomial(link = "logit"), data)
propensity <- predict(fit, type = "response") # P(D | Zs)
data$ps <- propensity

# check shared support of treatment and control on Propensity Scores
# overlapping hists plot
ggplot() +
  geom_histogram(data = data, mapping = aes(x= ps, fill = as.factor(d)), alpha = .7, position= "identity
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# percent of data unsupported
shared_suport <- sort(c(range(data[data$d == 1, "ps"]), range(data[data$d == 0, "ps"])))[2:3]
unsupported_rows <- (data$ps < shared_suport[1]) |  (data$ps > shared_suport[2])
(sum(unsupported_rows)/nrow(data)) * 100
```

```
## [1] 2.08
```

```
# 2.08% of data set is unsupported... pretty good!

# match treatments and controls
matched <- match(data, D_col = "d", PS_col = "ps")

# ATE, ATT, ATC
estimates <- get_estimates(matched)

estimates$ATE
```

```
## [1] 2.080637
```

```
 # 2.080637 ... very close to true value of 2!
estimates$ATT
```

```
## [1] 5.888378
```

```
# 5.888378 ... ok we believe this... close to the reg w/ cond estimate under these conditions
estimates$ATC
```

```
## [1] 1.712308
```

```
# 1.712308 .. close to two.. this shows that most of the data in the control


 ## conclusion: PSM preforms much better than regression with condition with a
# heavily imbalanced dataset, due to selection bias misleading the regression.
# Further we see that ATT only gives info on the treatment group that is selected with its selection bi
```

doing cond regression on a data matrix that is biased....

```
test <- lm(weighted_y ~ d + a + b, data = matched_data_att) # correct matched data?
summary(test)
```

```
##
## Call:
## lm(formula = weighted_y ~ d + a + b, data = matched_data_att)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##  -97.36  -16.06   -2.36    8.69 1888.07
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 101.7793     1.6856  60.381   <2e-16 ***
## d            -4.2998     2.0778  -2.069   0.0386 *
## a            15.1306     0.9922  15.250   <2e-16 ***
## b            13.4484     0.9960  13.502   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 57.75 on 3677 degrees of freedom
## Multiple R-squared:  0.09576,    Adjusted R-squared:  0.09502
## F-statistic: 129.8 on 3 and 3677 DF,  p-value: < 2.2e-16
```

```
# d estimate -2.6863



# why is ths so far from the true value of two 2.. bc it is based off a data matrix
# made up of only the y0_d1_hat approximated values gotten by nearest neighbors on
# propensity scores matched to y1_d1 treatments units... no pairing of control and
# approximated y1_d0_hat values... so the specification doesn't make theoretical
# sense as an estimation of ATE
diff <- predict(test, data.frame(d=c(1,0), a= mean(matched_data_att$a), b = mean(matched_data_att$b)))
# -2.686343
```

Is conditioning reggression or PSM more accurate??: balanced treatment and control

```
cond_reg_ATEs <- list()
PSM_ATEs <- list()
sample_ATE <- list()
for (n in 1:500) {
```

```r
  ## Data generation
  N = 5000
  a = rnorm(N)
  b = rnorm(N)

  # True Treatment effect = 2 (i.e. 102-100, the other parts just add noise of
  # differing variance)
  y1 = 102 + 6*a + 4*b + rnorm(N)
  y0 = 100 + 3*a + 2*b + rnorm(N)


  u = (a+b)/2

  p_d_given_a_b = plogis(u)
  d = rbinom(rep(1,N), 1, p_d_given_a_b)

  y = d * y1 + (1-d) * y0

  data = data.frame(d, y, a, b, u)

  # record sample ATE
  sample_ATE[n] <- mean(y1-y0)

  ## get conditioning regression ATEs
  cond_model <- lm(y ~ d+a+b, data)
  cond_reg_ATEs[n] <- cond_model$coefficients[2]

  ## Get PSM ATEs
  # make sure to read in the match() function from above first

  # make propensity scores
  fit <-  glm(d~a+b, family = binomial(link = "logit"), data)
  propensity <- predict(fit, type = "response") # P(D | Zs)
  data$ps <- propensity

  # match treatments and controls
  test <- match(data, D_col = "d", PS_col = "ps")
  all <- rbind(test$treated[,1:6], test$control[,1:6])
  all$y1 <- all$y
  all$y0 <- all$y
  all[all$d ==1, "y0"] <- test$treated$matched_controls_y
  all[all$d ==0, "y1"] <- test$control$matched_treateds_y

  # get psm ATE
  PSM_ATEs[n] <- mean(all$y1 - all$y0)




}

mean(unlist(sample_ATE))
```

```
## [1] 2.002899
```

```
#2nd 2.004365

mean(unlist(PSM_ATEs))
```

```
## [1] 2.007392
```

```
# 2.007392 #2nd 2.010691
sd(unlist(PSM_ATEs))
```

```
## [1] 0.06748241
```

```
# 0.06748241 #2nd 0.06430236

mean(unlist(cond_reg_ATEs))
```

```
## [1] 2.001612
```

```
# 2.001612 #2nd 2.004621
sd(unlist(cond_reg_ATEs))
```

```
## [1] 0.06747916
```

```
# 0.06747916 #2nd 0.06216361

# So it seems that over this sample there is slightly less bias in the conditioning
# regression appoach and pretty much identical variance...


mean(unlist(sample_ATE) - unlist(PSM_ATEs))
```

```
## [1] -0.004493538
```

```
#2nd -0.006326656
sd(unlist(sample_ATE) - unlist(PSM_ATEs))
```

```
## [1] 0.03757073
```

```
#2nd 0.03475738
#MSE
var(unlist(sample_ATE) - unlist(PSM_ATEs)) + mean(unlist(sample_ATE) - unlist(PSM_ATEs))^2
```

```
## [1] 0.001431751
```

```
#2nd MSE 0.001248102

mean(unlist(sample_ATE) - unlist(cond_reg_ATEs))
```

```
## [1] 0.001286897
```

```
#2nd -0.0002565596
sd(unlist(sample_ATE) - unlist(cond_reg_ATEs))
```

```
## [1] 0.04392708
```

```
# 0.04312974
var(unlist(sample_ATE) - unlist(cond_reg_ATEs)) + mean(unlist(sample_ATE) - unlist(cond_reg_ATEs))^2
```

```
## [1] 0.001931245
```

```
#2nd MSE  0.00186024 ... but now this way of calculating MSE says PSM has lower
#MSE..? essentially bc more variance...
```