Data Generation

Amy Pitts

2/8/2022

Data Generation

The method of data generation is inspired by the Austin & Small paper.

Covariate Generation

```
set.seed(8160) # once we do simulations this will need to change with interations

N <- 1000 # size of simulated dataset

x1 <- rnorm(N, 0, 1)
x2 <- rnorm(N, 0, 1)
x3 <- rnorm(N, 0, 1)
x4 <- rnorm(N, 0, 1)
x5 <- rnorm(N, 0, 1)
x6 <- rnorm(N, 0, 1)
x7 <- rnorm(N, 0, 1)
x8 <- rnorm(N, 0, 1)
x9 <- rnorm(N, 0, 1)
x10 <- rnorm(N, 0, 1)

beta_low <- log(1.25)
beta_med <- log(1.50)
beta_high <- log(1.75)
beta_Vhigh <- log(2)

error <- rnorm(N, 0, 3)

# Value gathered (Need to work on this)
beta_0_treat <- 1 # the intercept in the treament-selection model. This will detemrine the prevalance o

beta_0_outcome <- 0 # the intercept in the binary-outcome generating model
# its value will determine the incidence of the outcome
# the approaiate value of the intercept can be found using a bisection approach

beta_effect <- 1 # the log-odds ratio for the effect of treatment on the outcome that will be induce th

data_gen <- tibble(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10)
head(data_gen)
```

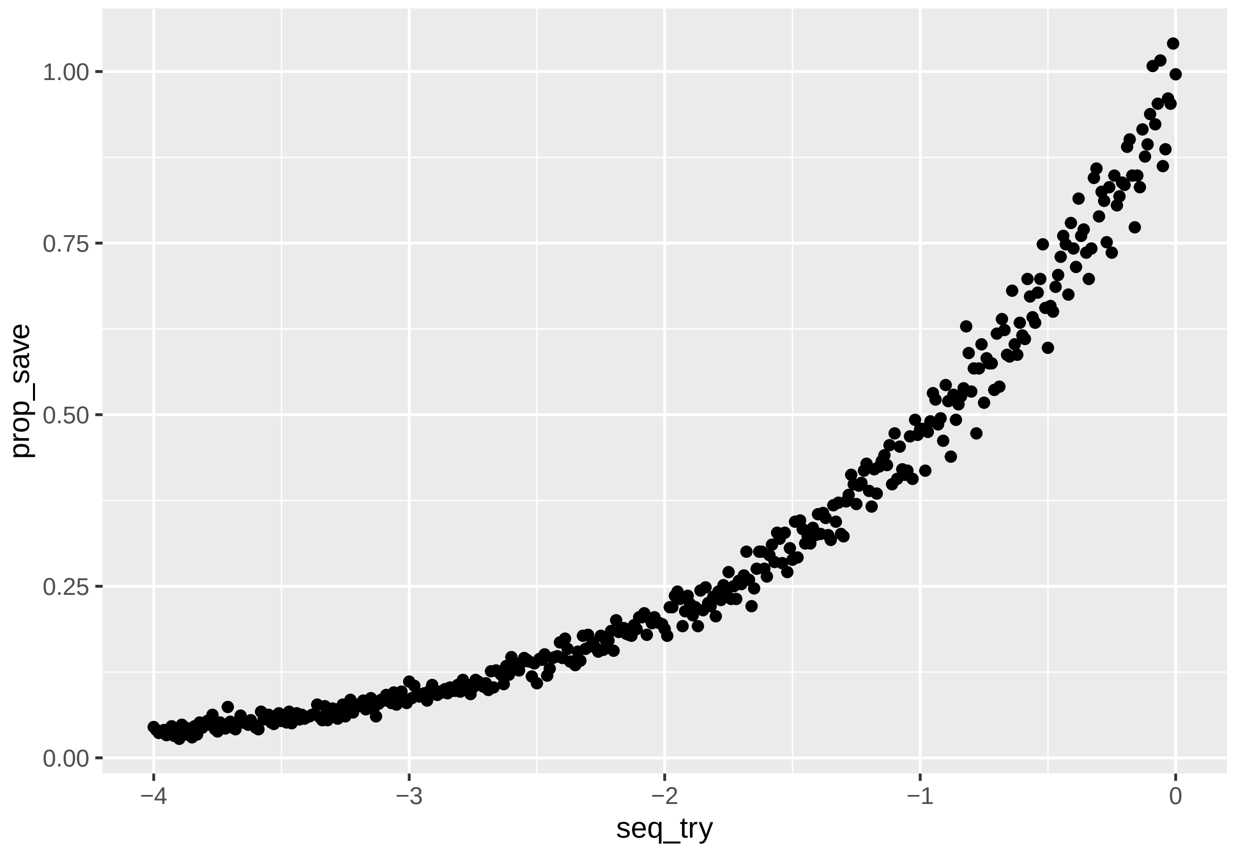
```
## # A tibble: 6 x 10
##        x1      x2     x3       x4      x5      x6     x7     x8       x9    x10
##     <dbl>   <dbl>  <dbl>    <dbl>   <dbl>   <dbl>  <dbl>  <dbl>    <dbl>  <dbl>
## 1 -2.31    2.01   -1.55  -0.286    1.22   -0.149  -1.14  -0.504  0.147   1.59
## 2 -0.916   0.177  -0.183 -0.606   -0.604  -0.0428  0.539 -2.51  -0.628  -0.326
## 3 -0.275   0.353  -0.123  0.00100 -2.17    1.20   -0.767  0.205  0.925   0.269
## 4  0.220   0.101  -1.16   0.455   -0.266  -0.0206  0.464  0.234 -0.0319 -0.453
## 5 -0.0215 -0.0631 -0.510  0.441    0.540   0.0416  0.147  0.772 -0.554   0.878
## 6 -0.131  -0.985   0.391 -0.737   -0.0864 -1.83   -0.218 -0.157  0.0961  0.720
```

treatment status

Determining the beta_0_treat value

Want a value that is close to 20% treatment selection

```
desired_prop = 0.2
seq_try = seq(-4,0, 0.01)
prop_save = rep(0, length(seq_try))
seq_select = tibble(seq_try, prop_save)
count = 0
for(i in seq_try) {
beta_0_treat = i
count = count + 1
treat_logit <- beta_0_treat+ beta_low*x1 + beta_med*x2 + beta_high*x3 + beta_low*x4 +
  beta_med*x5 + beta_high*x6 + beta_Vhigh*x7

p_treat <- exp(treat_logit)/(1+ exp(treat_logit))

treat <- rbinom(N, 1, p_treat)

treat_nontreat = tibble(treat) %>%
  group_by(treat) %>%
  summarize(num = n())

prop = treat_nontreat %>% filter(treat == 1 ) %>% pull(num) /
        treat_nontreat %>% filter(treat == 0 ) %>% pull(num)

seq_select$prop_save[count] =  prop
}
seq_select %>%
  ggplot(aes(x=seq_try, y=prop_save))+
  geom_point()
```



```
beta_0_treat = seq_select %>%
  filter(prop_save < desired_prop + 0.01) %>%
  filter(prop_save > desired_prop - 0.01) %>%
  summarise(
    avg = mean(seq_try)
  ) %>% pull(avg)
beta_0_treat
```

```
## [1] -2.014286
```

Now creating the treatment status

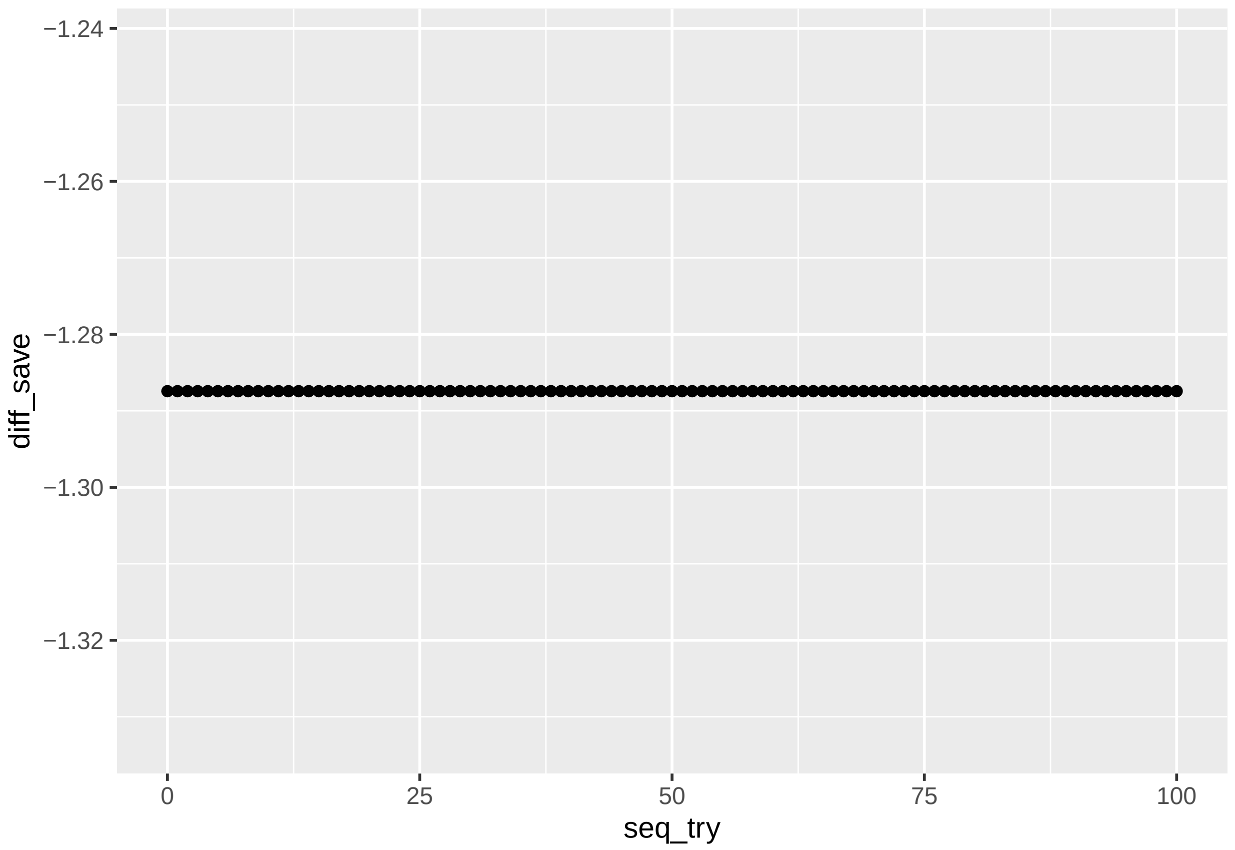
```
treat_logit <- beta_0_treat+ beta_low*x1 + beta_med*x2 + beta_high*x3 + beta_low*x4 +
  beta_med*x5 + beta_high*x6 + beta_Vhigh*x7

p_treat <- exp(treat_logit)/(1+ exp(treat_logit))

treat <- rbinom(N, 1, p_treat)

data_gen <- data_gen %>% mutate ( treat = treat)

table(treat)
```

```
## treat
##   0   1
## 857 143
```

Continous Y

determining beta_0_outcome

```r
seq_try = seq(0, 100, 1)
seq_select = tibble(seq_try, diff_save = rep(0, length(seq_try)))
count = 0
for(i in seq_try) {
  beta_0_outcome = i
  count = count + 1
  y_continous <- beta_0_outcome + 1*treat + beta_low*x4 + beta_med*x5 +
    beta_high*x6 + beta_Vhigh*x7 +  beta_low*x8 + beta_med*x9 +
    beta_high*x10 + error

  data_gen$y_continous = y_continous
  data_gen_save = data_gen  %>%
    group_by(treat) %>%
    summarize(
      mean_effect = mean(y_continous),
      sd_effect = sd(y_continous)
    )
  diff = data_gen_save %>% filter(treat == 0 ) %>% pull(mean_effect) -
          data_gen_save %>% filter(treat == 1 ) %>% pull(mean_effect)
  seq_select$diff_save[count] =  diff
}
seq_select %>%
  ggplot(aes(x=seq_try, y=diff_save))+
  geom_point()
```



doesn't seem to matter what the intercept is. the 1 is what is controlling the difference in group. I am not sure what is beta.0.outcome then…

Determining y using beta_0_outcome

```r
# continous outcome for each subject
beta_0_outcome = 0
y_continous <- beta_0_outcome+ 1*treat + beta_low*x4 + beta_med*x5 + beta_high*x6 + beta_Vhigh*x7 +
   beta_low*x8 + beta_med*x9 + beta_high*x10 + error

data_gen$y_continous = y_continous

# 1 is the treatment effect
data_gen  %>%
  group_by(treat) %>%
  summarize(
    mean_effect = mean(y_continous),
    sd_effect = sd(y_continous)
  )
```

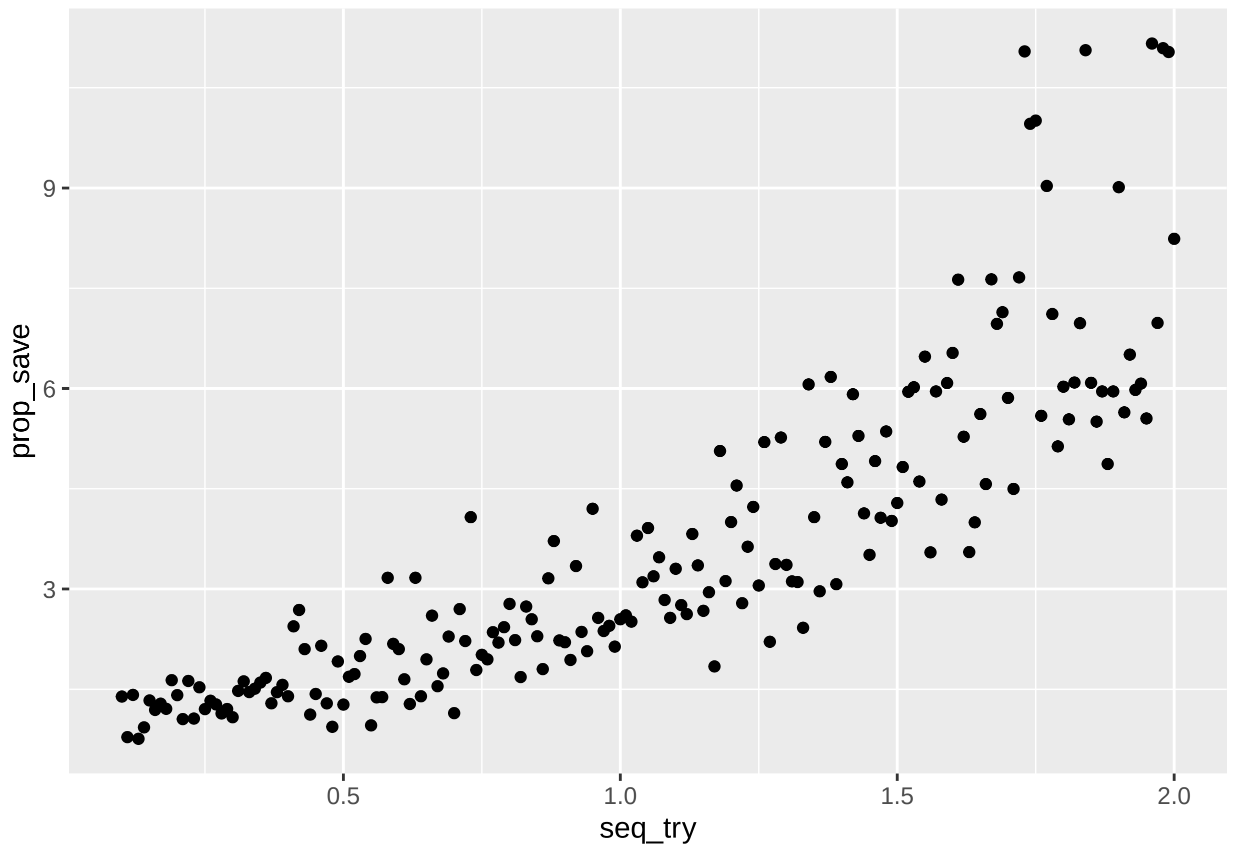
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
##   treat mean_effect sd_effect
##   <int>       <dbl>     <dbl>
## 1     0      -0.178      3.24
## 2     1       1.11       3.37
```

binary outcome

Finding the right beta_effect estimate.

```r
seq_try = seq(0.1,2, 0.01)
prop_save = rep(0, length(seq_try))
seq_select = tibble(seq_try, prop_save)
count = 0
for(i in seq_try) {
  beta_effect = i
  count = count + 1

  # binary outcome
  y_logit <- beta_effect*treat + beta_low*x4 + beta_med*x5 + beta_high*x6 +
    beta_Vhigh*x7 +beta_low*x8 + beta_med*x9 + beta_high*x10
  # don't need error here. error comes from rbinom

  p_outcome <- exp(y_logit)/(1+ exp(y_logit))
  y_binary <- rbinom(N, 1, p_outcome)

  data_gen$y_binary = y_binary

  table_prop = data_gen %>%
    mutate(treat = ifelse(treat == 1, "treated", "nontreated")) %>%
    group_by(treat, y_binary) %>%
    summarize(n=n()) %>%
    pivot_wider(
      names_from = treat,
      values_from = n
    )

  prop = (table_prop %>% filter(y_binary == 1) %>% pull(treated)/
       table_prop %>% filter(y_binary == 0) %>% pull(treated))-
    (table_prop %>% filter(y_binary == 1) %>%  pull(nontreated) /
       table_prop %>% filter(y_binary == 0) %>% pull(nontreated))

  seq_select$prop_save[count] = prop
}

seq_select %>%
  ggplot(aes(x=seq_try, y=prop_save))+
  geom_point()
```



```
desired_prop = 1
beta_effect = seq_select %>%
  filter(prop_save < desired_prop + 0.01) %>%
  filter(prop_save > desired_prop - 0.01) %>%
  summarise(
    avg = mean(seq_try)
  ) %>% pull(avg)
beta_effect
```

```
## [1] NaN
```

Using that estimate.

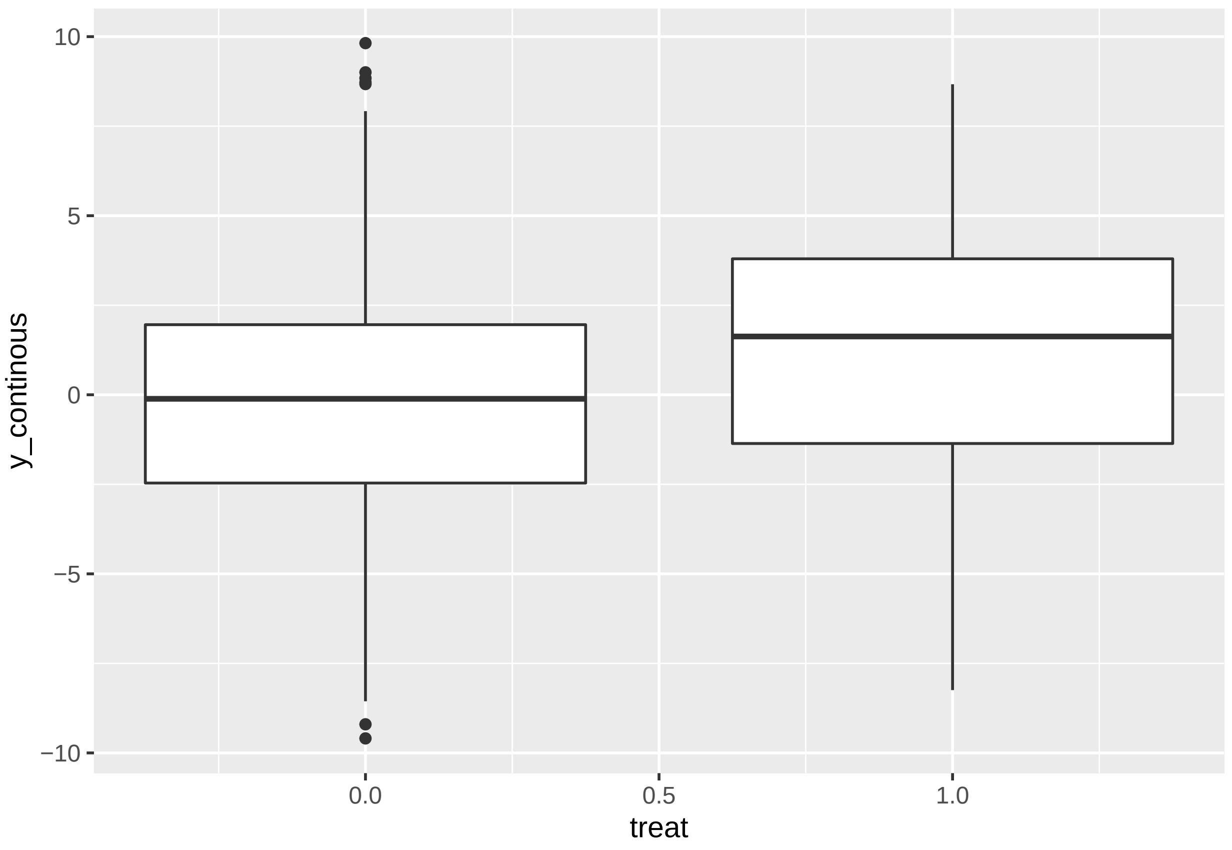
```
# binary outcome
y_logit <- beta_effect*treat + beta_low*x4 + beta_med*x5 + beta_high*x6 +
  beta_Vhigh*x7 +beta_low*x8 + beta_med*x9 + beta_high*x10 # don't need error here.

p_outcome <- exp(y_logit)/(1+ exp(y_logit))
y_binary <- rbinom(N, 1, p_outcome)
```

```
## Warning in rbinom(N, 1, p_outcome): NAs produced
```

```
data_gen$y_binary = y_binary
```

Evaulating our Methods

```
# treated vs non treated
data_gen  %>%
  ggplot(aes(y=y_continous, x=treat, group=treat)) +
  geom_boxplot()
```



```
table(data_gen$treat)
```

```
##
##   0   1
## 857 143
```

```
# contious
data_gen  %>%
  group_by(treat) %>%
  summarize(
    mean_effect = mean(y_continous),
    sd_effect = sd(y_continous)
  )
```

```
## # A tibble: 2 x 3
##   treat mean_effect sd_effect
##   <int>       <dbl>     <dbl>
## 1     0      -0.178      3.24
## 2     1       1.11       3.37
```

```
# binary
table_prop = data_gen %>%
  mutate(treat = ifelse(treat == 1, "treated", "nontreated")) %>%
  group_by(treat, y_binary) %>%
  summarize(n=n()) %>%
  pivot_wider(
    names_from = treat,
    values_from = n
  )

(table_prop %>% filter(y_binary == 1) %>% pull(nontreated) /
table_prop %>% filter(y_binary == 0) %>% pull(nontreated))-
(table_prop %>% filter(y_binary == 1) %>% pull(treated)/
table_prop %>% filter(y_binary == 0) %>% pull(treated))
```

```
## numeric(0)
```

Matching!

I am reading off of link

Pre-analysis using non-matched data Continous Y

```
data_gen %>%
  group_by(treat) %>%
  summarise(n = n(),
            mean_math = mean(y_continous),
            std_error = sd(y_continous) / sqrt(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 4
##   treat     n mean_math std_error
##   <int> <int>     <dbl>     <dbl>
## 1     0   857    -0.178     0.111
## 2     1   143     1.11      0.282
```

```
with(data_gen, t.test(y_continous ~ treat))
```

```
##
##  Welch Two Sample t-test
##
## data:  y_continous by treat
## t = -4.2526, df = 188.59, p-value = 3.323e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.8846249 -0.6902325
## sample estimates:
## mean in group 0 mean in group 1
##      -0.1782865       1.1091421
```

Difference-in-means: pre-treatment covariates

```
data_gen_cov <- c('x4', 'x5', 'x6', 'x7')
data_gen %>%
  group_by(treat) %>%
  select(one_of(data_gen_cov)) %>%
  summarise_all(funs(mean(., na.rm = T)))
```

```
## Adding missing grouping variables: `treat`
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```
## # A tibble: 2 x 5
##   treat      x4      x5      x6      x7
##   <int>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     0 -0.0331 -0.0945 -0.0686 -0.0494
## 2     1  0.243   0.312   0.383   0.498
```

Propensity score estimation

We estimate the propensity score by running a logit model (probit also works) where the outcome variable is a binary variable indicating treatment status. What covariates should you include? For the matching to give you a causal estimate in the end, you need to include any covariate that is related to both the treatment assignment and potential outcomes. I choose just a few covariates below—they are unlikely to capture all covariates that should be included. You'll be asked to come up with a potentially better model on your own later.

```
m_ps <- glm(treat ~ x4 + x5 + x6 + x7 ,
            family = binomial(), data = data_gen)
summary(m_ps)
```

```
##
## Call:
## glm(formula = treat ~ x4 + x5 + x6 + x7, family = binomial(),
##     data = data_gen)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3643  -0.5763  -0.4107  -0.2659   2.9380
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.11066    0.11518 -18.324  < 2e-16 ***
## x4           0.32317    0.09837   3.285  0.00102 **
## x5           0.47333    0.09825   4.818 1.45e-06 ***
## x6           0.48349    0.09650   5.010 5.44e-07 ***
## x7           0.67645    0.10461   6.467 1.00e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 820.74  on 999  degrees of freedom
## Residual deviance: 720.58  on 995  degrees of freedom
## AIC: 730.58
##
## Number of Fisher Scoring iterations: 5
```

Using this model, we can now calculate the propensity score for each student. It is simply the student's predicted probability of being Treated, given the estimates from the logit model. Below, I calculate this propensity score using predict() and create a dataframe that has the propensity score as well as the student's actual treatment status.

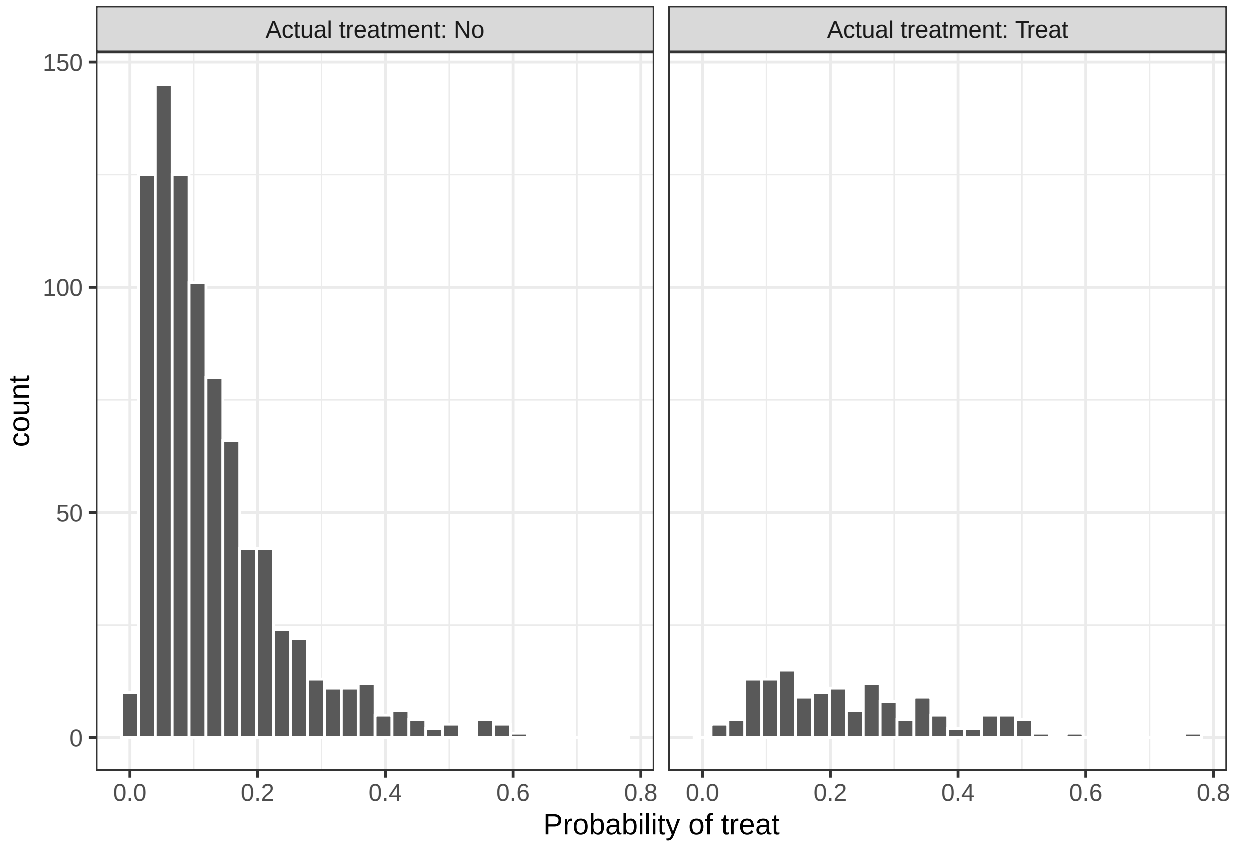
```r
prs_df <- data.frame(pr_score = predict(m_ps, type = "response"),
                     treat = m_ps$model$treat)
head(prs_df)
```

```
##     pr_score treat
## 1 0.07816243     0
## 2 0.09547404     0
## 3 0.04410302     0
## 4 0.14361969     1
## 5 0.16899806     0
## 6 0.03165497     0
```

Evaulating

```r
labs <- paste("Actual treatment:", c("Treat", "No"))
prs_df %>%
  mutate(treat = ifelse(treat == 1, labs[1], labs[2])) %>%
  ggplot(aes(x = pr_score)) +
  geom_histogram(color = "white") +
  facet_wrap(~treat) +
  xlab("Probability of treat") +
  theme_bw()
```

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



Executing a matching algorithm

```r
library(MatchIt)
```

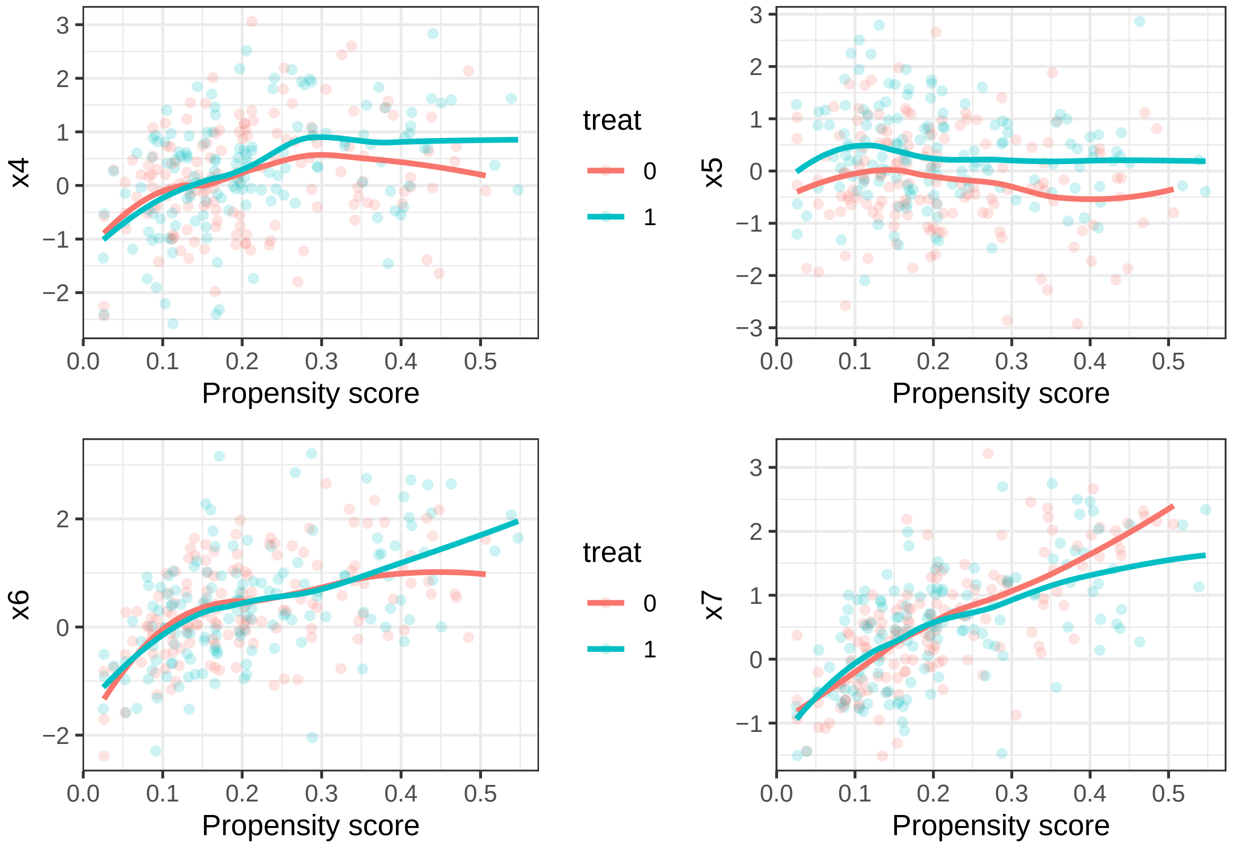
```
## Warning: package 'MatchIt' was built under R version 4.0.5
```

```r
data_gen_nomiss <- data_gen %>%  # MatchIt does not allow missing values
  select(y_continous, treat, one_of(data_gen_cov)) %>%
  na.omit()

mod_match <- MatchIt::matchit(treat ~ x4 + x4 + x6 +  x7,
                              method = "nearest",
                              data = data_gen_nomiss)
dta_m <- match.data(mod_match)
dim(dta_m)
```

```
## [1] 286   9
```

Evaulate

```r
fn_bal <- function(dta, variable) {
  dta$variable <- dta[, variable]
  dta$treat <- as.factor(dta$treat)
  support <- c(min(dta$variable), max(dta$variable))
  ggplot(dta, aes_string(x = "distance", y = variable, color = "treat")) +
    geom_point(alpha = 0.2, size = 1.3) +
    geom_smooth(method = "loess", se = F) +
    xlab("Propensity score") +
    ylab(variable) +
    theme_bw() +
    ylim(support)
}

library(gridExtra)
grid.arrange(
   fn_bal(dta_m, "x4"),
   fn_bal(dta_m, "x5") + theme(legend.position = "none"),
   fn_bal(dta_m, "x6"),
   fn_bal(dta_m, "x7") + theme(legend.position = "none"),
   nrow = 2, widths = c(1, 0.8)
)
```



```r
dta_m %>%
  group_by(treat) %>%
  select(one_of(data_gen_cov)) %>%
  summarise_all(funs(mean))
```

```
## # A tibble: 2 x 5
##   treat    x4     x5    x6    x7
##   <int> <dbl>  <dbl> <dbl> <dbl>
## 1     0 0.155 -0.187 0.377 0.524
## 2     1 0.243  0.312 0.383 0.498
```