Introduction to IP Weighting Using R

Ashley I Naimi

Spring 2024

Contents

1 Introduction to Propensity Score Methods

So far in this course, we've focused on obtaining quantitative estimates of an exposure effect using an outcome modeling approach. In this approach, one regresses the outcome against the exposure and confounders. This works for the conditionally adjusted estimator, where we read the coefficient for the exposure from the regression model, or the marginally adjusted estimator (e.g., using marginal standardization). The basic formulation of this outcome modeling approach can be depicted heuristically using the DAG in Figure 1:

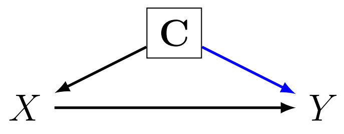


Figure 1: Directed acyclic graph depicting confounder adjustment using an outcome modelling approach. In this approach, information from the confounder to the outcome is 'blocked' (blue arrow). With this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome association unconfounded.

In this Figure, the open back-door path from X to Y is blocked by conditioning on C. This leads to a conditionally adjusted estimate of the exposure effect. One can marginalize over the distribution on C to get a marginally adjusted estimate from a model for the outcome. For a binary outcome, such estimates may be obtained on the risk difference, risk ratio, or odds ratio scales, or one may obtain a marginally adjusted estimate of the cumulative risk function that would be observed if everyone were exposed or unexposed.

On the other hand, we may want to obtain an adjusted estimate of the exposure effect by modeling the exposure. This may be the case when the event under study (e.g., mortality, birth defect, heart attack) is very rare in the sample, and there are a large number of confounders that we must adjust for. In this case, it may be preferable to limit the number of confounders in the outcome model, and adjust for confounding by modeling the exposure.

This can be accomplished using propensity score methods.

The propensity score was first defined by Rosenbaum and Rubin (Rosenbaum and Rubin, 1983) as the conditional probability of receiving the treatment (or, equivalently, of being exposed). The true propensity score is defined as

$$e(C) = P(X = 1 \mid C)$$

This propensity score is typically known in a randomized trial. It's important to distinguish this true PS from the estimated PS:

$$\hat{e}(C) = \hat{P}(X = 1 \mid C)$$

This estimated PS can be fit using a parametric model, in which case, we might write[1]:

$$\hat{e}(C; \alpha) = \hat{P}(X = 1 \mid C) = \text{expit}(\alpha C)$$

[1] Recall, the $\text{expit}(a)$ function is the inverse of the $\text{logit}(a)$ function, defined as: $1/[1 + exp(-a)]$

For reasons that are not entirely obvious, it is usually better to use the estimated PS, even when the true PS is known (Robins et al., 1992, Henmi and Eguchi (2004)). For example, in the context of a randomized trial, the true propensity score is actually known exactly, and is equal to the probability of receiving treatment. However, the performance of a propensity score based estimator (in terms of statistical efficiency, which determines, among other things, the size of the standard errors) will be better after using an estimated propensity score, even in this setting.

If the set of conditioning variables consists of the relevant confounding variables, the propensity score can be used to invoke conditional exchangeability. To see why, consider the case where the probability of being exposed is conditional on two binary confounders:

$$f(C) = P(X = 1 \mid C) = \text{expit}(\alpha_0 + \alpha_1 C_1 + \alpha_2 C_2 + \alpha_3 C_1 C_2)$$

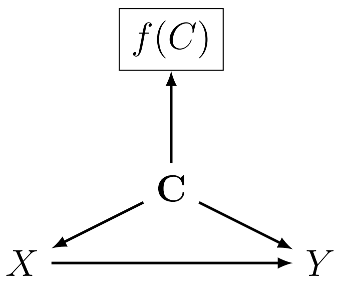
Consider that, for two binary confounders, there are four possible joint confounder levels:

C1	C2	$P(X = 1 \mid C)$
0	0	$\text{expit}(\alpha_0)$
1	0	$\text{expit}(\alpha_0 + \alpha_1)$
0	1	$\text{expit}(\alpha_0 + \alpha_2)$
1	1	$\text{expit}(\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3)$

Notice also that there are four parameters in the above model, one parameter for each level. This implies that, for this simple example, there is a unique propensity score value for each unique confounder level. In other words, the one-dimensional propensity score in this example contains all the information available in the two-dimensional set of confounders. We can thus reduce the complexity of the set of confounders to a single variable, and adjust for this variable instead of the confounders. For example:

$$E[Y \mid X, f(C)] = \beta_0 + \beta_1 X + \beta_c f(C)$$

Heuristically (again), we can show why this approach works using the DAG in Figure 2



$$\begin{tikzcd} & \boxed{f(C)} & \\ & \mathbf{C} \arrow[u] \arrow[dl] \arrow[dr] & \\ X \arrow[rr] & & Y \end{tikzcd}$$

Figure 2: Directed acyclic graph depicting confounder adjustment using an outcome modelling approach. In this approach, information from the confounder to the outcome is 'blocked' (blue arrow). With this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome association unconfounded.

In this DAG, $f(C)$ is a proxy for all the variables C. Thus, we can use $f(C)$ to replace C in an outcome regression model.

As a univariate proxy for the multivariate set of confounders, we can use the propensity score to adjust for confounding using a number of different techniques. These techniques include regression adjustment, propensity score matching, stratification, and weighting. In this lecture, we will quickly cover the first three and focus on the creation and use of IP-weights. The reason for this emphasis is that (i) in epidemiology, the most commonly used PS technique is inverse probability weighting, and so it is important to know how to implement; and (ii), there are theoretical justifications suggesting that weighting is the most optimal use of the PS.

2 Inverse Probability Weighting: Intuition

The most commonly employed propensity score adjustment technique is inverse probability weighting. The simple heuristic often used to describe the way IP-weighting works is that, when applied to data, they yield a "pseudo population" where there is no longer an effect of the confounder on the exposure. The causal structure of the variables in this new pseudo-population can be depicted in Figure 3:

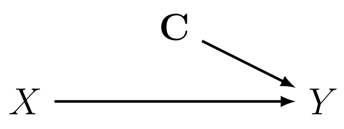


Figure 3: Directed acyclic graph depicting the causal relations between variables in a 'pseudo-population' obtained via inverse probability weighting. Again, with this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome association unconfounded.

The weights for each individual needed to create this pseudo-population are defined as the inverse of the probability of receiving their observed exposure. Let's consider the following simple example to explain why this works. In this example, there are 20 observations, with one binary confounder (50:50 split) and one binary exposure. Let's suppose the probability that the exposure `x = 1` is 0.2 for those with `c = 0` and 0.9 for those with `c = 1`:

Under these scenarios, we'd have exposure and confounder data that looks like this:

```r
c <- c(0, 0, 1, 1)
x <- c(1, 0, 1, 0)
n <- c(2, 8, 9, 1)

tibble(x, c, n)
```

```
## # A tibble: 4 x 3
##       x     c     n
##   <dbl> <dbl> <dbl>
## 1     1     0     2
## 2     0     0     8
## 3     1     1     9
## 4     0     1     1
```

Let's focus on the stratum of individuals with `c = 0`. In this stratum, there are 2 exposed individuals, and 8 unexposed individuals. Now let's ask what these data should look like if we were able to implement an ideal (marginally) randomized trial with the probability of treatment being 50% for all individuals. We would expect that within the stratum of individuals with `c = 0`, there would be 5 exposed and 5 unexposed individuals. Inverse probability weighting seeks to accomplish this balance.

Consider that the inverse probability of the **observed exposure** among those with `c = 0` is 0.2 for those who were actually exposed, and 0.8 for those who were unexposed.

The inverse probability weight is thus $\frac{1}{0.2} = 5$ for the exposed in this confounder stratum, and $\frac{1}{0.8} = 1.25$ for the unexposed in this confounder stratum.

This suggests that, in their contribution to the overall analysis, the two exposed individuals in the `c = 0` status would each receive a weight of 5, while the eight unexposed individuals in the `c = 0` stratum would each receive a weight of 1.25. Under these conditions, we would have a re-balanced set of observations in the `c = 0` stratum of:

$$\text{Exposed Observations:} \quad 5 \times 2 = 10$$

$$\text{Unexposed Observations:} \quad 8 \times 1.25 = 10$$

In effect, our inverse probability weighting strategy made it such that we now have an equal number of exposed and unexposed observations within the `c = 0` stratum. In these weighted data, we can now compute the difference in the outcome among the exposed and unexposed individuals (if we had it) to obtain an estimate of our average treatment effect.

3 Inverse Probability Weighting In Practice

Simply taking the inverse of the probability of the observed exposure, while valid, is not the usual strategy for implementing inverse probability weights. In practice, one will often use stabilized inverse probability weights, or stabilized normalized inverse probability weights.

At times, there may be a reason to “truncate” or, more accurately, trim the weights to reduce the impact of potential outliers in the weights.[2]

[2] In contrast to our emphasis of the usage of the word “truncation” which refers to the removal of observations from the dataset, researchers will often refer to “truncating” the weights, which sets the largest value to be equal to the 99th or 95th percentile values. This is more accurately referred to as “trimming” the weights, since no truncation is occurring.

Furthermore, it’s important to note that “data” are often not weighted, but rather the contribution that each individual in the sample makes to the estimating function (e.g., likelihood, estimating equation, or other function used to find parameters). This is important in that one must choose a fitting algorithms that allows for this type of weighting.

To start, the simplest type of weight used in practice is the stabilized inverse probability weight. These are often defined as:

$$sw = \begin{cases} \dfrac{P(X=1)}{P(X=1 \mid C)} & \text{if } X = 1 \\ \dfrac{P(X=0)}{P(X=0 \mid C)} & \text{if } X = 0 \end{cases}$$

but are sometimes written more succinctly as[3]:

$$sw = \frac{f(X)}{f(X \mid C)}$$

[3] This formulation is unusual, since $f(.)$ represents the probability density function, which is usually taken for a specific realization of the random variable. However, in this case, because the weights are defined as a function of the observed exposure status, the argument in the operator is the observed data (denoted with capital letter), as opposed to some specific realization.

Let's use the cohort data again to construct the stabilized weights. We will re-fit the PS model to the data, and construct the weights:

```r
file_loc <- url("https://bit.ly/47ECRcs")

#' This begins the process of cleaning and formatting the data
nhefs <- read_csv(file_loc) %>%
    select(qsmk, wt82_71, sex, age, exercise,
        race, income, marital, school, asthma,
        bronch, starts_with("alcohol"), -alcoholpy,
        starts_with("price"), starts_with("tax"),
        starts_with("smoke"), smkintensity82_71) %>%
    mutate(wt_delta = as.numeric(wt82_71 >
        median(wt82_71, na.rm = T)), income = as.numeric(income >
        15), marital = as.numeric(marital >
        2), alcoholfreq = as.numeric(alcoholfreq >
        1)) %>%
    na.omit(.)

nhefs$id <- 1:nrow(nhefs)
```

```r
# create the propensity score in the
# dataset
nhefs$propensity_score <- glm(qsmk ~ exercise +
    sex + age + race + income + marital +
    school + asthma + bronch + alcoholfreq,
    data = nhefs, family = binomial("logit"))$fitted.values

# stabilized inverse probability
# weights
nhefs$sw <- (mean(nhefs$qsmk)/nhefs$propensity_score) *
    nhefs$qsmk + ((1 - mean(nhefs$qsmk))/(1 -
    nhefs$propensity_score)) * (1 - nhefs$qsmk)

summary(nhefs$sw)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4523  0.8979  0.9764  1.0006  1.0688  3.5141
```

```r
nhefs %>%
    select(id, wt_delta, qsmk, sex, age,
        exercise, propensity_score, sw) %>%
    print(n = 5)
```

```
## # A tibble: 1,055 x 8
##      id wt_delta  qsmk   sex   age exercise propensity_score    sw
##   <int>    <dbl> <dbl> <dbl> <dbl>    <dbl>            <dbl> <dbl>
## 1     1        0     0     0    42        2            0.118 0.853
## 2     2        1     0     0    36        0            0.168 0.904
## 3     3        1     0     0    68        2            0.208 0.950
## 4     4        1     0     0    40        1            0.295 1.07
## 5     5        1     0     1    43        1            0.106 0.842
## # i 1,050 more rows
```

As we can see from the output above, the stabilized weights are, in fact, well behaved, with a mean of one and a max value that is small.

```r
model_RD_weighted <- glm(wt_delta ~ qsmk,
    data = nhefs, weights = sw, family = quasibinomial("identity"))

summary(model_RD_weighted)$coefficients
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 0.4792921 0.01775233 26.998830 1.985493e-122
## qsmk        0.1499691 0.03476671  4.313584  1.757278e-05
```

At times, the mean and max of the stabilized weights are sub-optimal, in that the mean may not be one, or the max may be too large for comfort. One strategy we can use here is to normalize the weights, by dividing the stabilized weights by the max stabilized weight:

```r
nhefs$sw_norm <- nhefs$sw/max(nhefs$sw)

summary(nhefs$sw_norm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1287  0.2555  0.2779  0.2847  0.3041  1.0000
```

```r
nhefs %>%
    select(id, wt_delta, qsmk, sex, age,
        exercise, propensity_score, sw, sw_norm) %>%
    print(n = 5)
```

```
## # A tibble: 1,055 x 9
##      id wt_delta  qsmk   sex   age exercise propensity_score    sw sw_norm
##   <int>    <dbl> <dbl> <dbl> <dbl>    <dbl>            <dbl> <dbl>   <dbl>
## 1     1        0     0     0    42        2            0.118 0.853   0.243
## 2     2        1     0     0    36        0            0.168 0.904   0.257
## 3     3        1     0     0    68        2            0.208 0.950   0.270
## 4     4        1     0     0    40        1            0.295 1.07    0.304
## 5     5        1     0     1    43        1            0.106 0.842   0.240
## # i 1,050 more rows
```

In this case, the mean of the normalized weights is no longer expected to be one. However, the max weight will be one by definition. These weights can then be used in the same way:

```
model_RD_weighted_norm <- glm(wt_delta ~
    qsmk, data = nhefs, weights = sw_norm,
    family = quasibinomial("identity"))

summary(model_RD_weighted_norm)$coefficients
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 0.4792921 0.01775233 26.998830 1.985493e-122
## qsmk        0.1499691 0.03476671  4.313584  1.757278e-05
```

Finally, instead of normalizing, more commonly, researchers will “trim” the weights to avoid problems induced by very large weights. The procedure for doing this requires simply replacing all weight values greater than a certain percentile with their percentile values:

```
quantile(nhefs$sw, 0.99)
```

```
##      99%
## 1.860241
```

```
nhefs <- nhefs %>%
    mutate(sw_trim = if_else(sw > quantile(sw,
        0.99), quantile(sw, 0.99), sw))

summary(nhefs$sw)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4523  0.8979  0.9764  1.0006  1.0688  3.5141
```

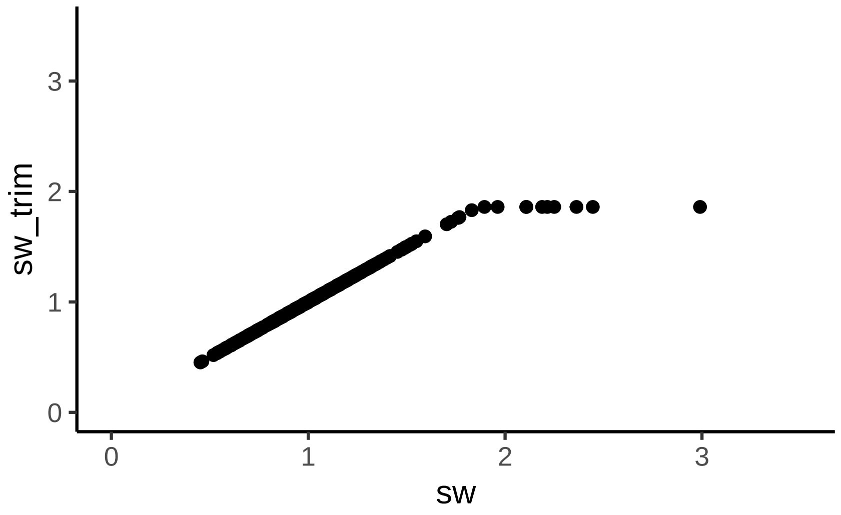
```
summary(nhefs$sw_trim)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4523  0.8979  0.9764  0.9953  1.0688  1.8602
```

We can see how this changes the values in the following plot:

```r
ggplot(nhefs) + geom_jitter(aes(sw, sw_trim)) +
    scale_x_continuous(limits = c(0, 3.5)) +
    scale_y_continuous(limits = c(0, 3.5))
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



When trimming weights, it's often best to start at the highest percentile (e.g., 99th percentile) and work your way down until the mean of the weights is close enough to 1 and the max weight is not too large. This strategy of starting with the highest percentile allows us to avoid the brunt of the potential bias that results from trimming. In fact, the key idea behind weight trimming is to optimize the bias-variance tradeoff being made (Cole and Hernán, 2008).

A final note that is important with these weighted approaches is to consider how to estimate the standard errors. In fact, the model-based standard errors are no longer valid when weighting is used. One must instead use the robust variance estimators, or the bootstrap.

In the previous set of notes, we saw how to use the `sandwich` function to implement the robust variance estimator. Here, we'll use the `coeftest` function in the `lmtest` package, which relies on the `sandwich` function without making an explicit call to the `sandwich` package. The `coeftest` function also allows us to obtain our results in a cleaner format. For example:

```r
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:data.table':
##
##     yearmon, yearqtr

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(sandwich)

coeftest(model_RD_weighted_norm, vcov. = vcovHC)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 0.479292   0.017860 26.8367 < 2.2e-16 ***
## qsmk        0.149969   0.037051  4.0477 5.173e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

One can then construct CIs in the standard way using the estimated standard error in the output above.

4 IP Weighting for Categorical and Continuous Exposures

Oftentimes, we're interested in estimating the effect of a categorical or continuous exposure using inverse probability weighting. Constructing weights for these types of exposures requires additional considerations.

Suppose, for example, we're interested in estimating the effect of exercise on greater than median weight change in the NHEFS data. Per the NHEFS codebook, the exercise variable is based on the following question regarding the amount of exercise in 1971, classified as:

- exercise = 0: "much exercise"
- exercise = 1: "moderate exercise"
- exercise = 2: "little or no exercise"

Suppose we were interested in computing two risk differences comparing "much exercise" to "little or no exercise", and comparing "moderate exercise" to "little or no exercise". These both represent average treatment effects, and we can compute these using IP weighting. To construct weights for a categorical exposure, we can use multinomial logistic regression to obtain the predicted probability of being in the **observed exposure category.** In R, multinomial regression can be deployed using the `VGAM` package via the `vglm` function. If we assume that sex, age, race, income, and education (school) are confounders of the exercise and weight change relation, we can fit the following propensity score model:

```
library(VGAM)

ps_mod <- vglm(factor(exercise) ~ sex + age + race + income + school,
               data = nhefs,
               family = "multinomial")

summary(ps_mod)
```

```
##
## Call:
## vglm(formula = factor(exercise) ~ sex + age + race + income +
##     school, family = "multinomial", data = nhefs)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept):1  0.402904   0.596525   0.675 0.499410
## (Intercept):2 -0.412281   0.482996  -0.854 0.393332
## sex:1         -0.666444   0.181738  -3.667 0.000245 ***
## sex:2         -0.013912   0.144787  -0.096 0.923450
## age:1         -0.035234   0.007887  -4.468 7.91e-06 ***
## age:2         -0.014325   0.006152  -2.328 0.019892 *
## race:1        -0.856622   0.283547  -3.021 0.002519 **
## race:2        -0.651444   0.207982  -3.132 0.001735 **
## income:1       0.567965   0.280502   2.025 0.042886 *
## income:2       0.291903   0.205496   1.420 0.155468
## school:1       0.043191   0.032963   1.310 0.190108
## school:2       0.098112   0.027702   3.542 0.000398 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])
##
## Residual deviance: 2135.574 on 2098 degrees of freedom
##
## Log-likelihood: -1067.787 on 2098 degrees of freedom
##
## Number of Fisher scoring iterations: 5
##
## No Hauck-Donner effect found in any of the estimates
##
##
## Reference group is level  3  of the response
```

Once this propensity score model is fit, we need to obtain predicted probabilities of being exposed in each category:

```
ps_matrix <- cbind(predict(ps_mod, type = "response"), nhefs$exercise)

ps_matrix <- data.frame(ps_matrix)

names(ps_matrix) <- c("pEx0","pEx1","pEx2","Observed_Exercise")
```

```r
head(ps_matrix)
```

```
##         pEx0      pEx1      pEx2 Observed_Exercise
## 1 0.18680740 0.2722355 0.5409571                 2
## 2 0.32452030 0.3792307 0.2962490                 0
## 3 0.05590035 0.1656755 0.7784242                 2
## 4 0.29567305 0.4192774 0.2850495                 1
## 5 0.06822032 0.2867693 0.6450104                 1
## 6 0.23966182 0.4048806 0.3554575                 2
```

Let's look at the summary distributions of each level:

```r
summary(ps_matrix[, 1])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03074 0.14934 0.20876 0.21043 0.26476 0.40426
```

```r
summary(ps_matrix[, 2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1170  0.3948  0.4436  0.4408  0.5051  0.6072
```

```r
summary(ps_matrix[, 3])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1523  0.2622  0.3219  0.3488  0.4076  0.8241
```

We can also explore the overlap in these propensity scores:

```r
plot_dat1 <- data.frame(ps_matrix[,c(1,4)], pEx = "pEx0")
plot_dat2 <- data.frame(ps_matrix[,c(2,4)], pEx = "pEx1")
plot_dat3 <- data.frame(ps_matrix[,c(3,4)], pEx = "pEx2")

names(plot_dat1) <- names(plot_dat2) <- names(plot_dat3) <- c("propensity_score",
                                                               "exposure_level",
"pEx")
```

```
plot_dat <- rbind(plot_dat1,
                  plot_dat2,
                  plot_dat3)

dim(plot_dat)
```

```
## [1] 3165    3
```

```
dim(ps_matrix)
```

```
## [1] 1055    4
```

```
head(ps_matrix, 10)
```

```
##          pEx0      pEx1      pEx2 Observed_Exercise
## 1  0.18680740 0.2722355 0.5409571                 2
## 2  0.32452030 0.3792307 0.2962490                 0
## 3  0.05590035 0.1656755 0.7784242                 2
## 4  0.29567305 0.4192774 0.2850495                 1
## 5  0.06822032 0.2867693 0.6450104                 1
## 6  0.23966182 0.4048806 0.3554575                 2
## 7  0.27922028 0.4215796 0.2992001                 2
## 8  0.27614023 0.4404681 0.2833917                 1
## 9  0.32585586 0.4306075 0.2435366                 2
## 10 0.15079941 0.5022256 0.3469750                 0
```

```
head(plot_dat, 10)
```

```
##    propensity_score exposure_level  pEx
## 1        0.18680740              2 pEx0
## 2        0.32452030              0 pEx0
## 3        0.05590035              2 pEx0
## 4        0.29567305              1 pEx0
## 5        0.06822032              1 pEx0
## 6        0.23966182              2 pEx0
## 7        0.27922028              2 pEx0
## 8        0.27614023              1 pEx0
## 9        0.32585586              2 pEx0
## 10       0.15079941              0 pEx0
```

```
ggplot(plot_dat) +
  geom_density(aes(x = propensity_score,
                   group = factor(exposure_level),
                   fill  = factor(exposure_level)),
               bw = .03, alpha = .25) +
  facet_wrap(~ pEx, ncol = 1) +
  scale_x_continuous(expand = c(0,0), limits = c(0,1)) +
  scale_y_continuous(expand = c(0,0))
```

```
ggsave(here("figures", "ps_overlap_3level.pdf"),
       width = 10,
       height = 16,
       units = "cm")
```

This is a propensity score overlap plot for the categorical (three level) exposure, and it does not suggest problematic overlap. If we move forward with the analysis, we need to obtain the predicted probability of the observed exposure from this propensity score matrix:

```
head(ps_matrix)
```

```
##         pEx0      pEx1      pEx2 Observed_Exercise
## 1 0.18680740 0.2722355 0.5409571                 2
## 2 0.32452030 0.3792307 0.2962490                 0
## 3 0.05590035 0.1656755 0.7784242                 2
## 4 0.29567305 0.4192774 0.2850495                 1
## 5 0.06822032 0.2867693 0.6450104                 1
```

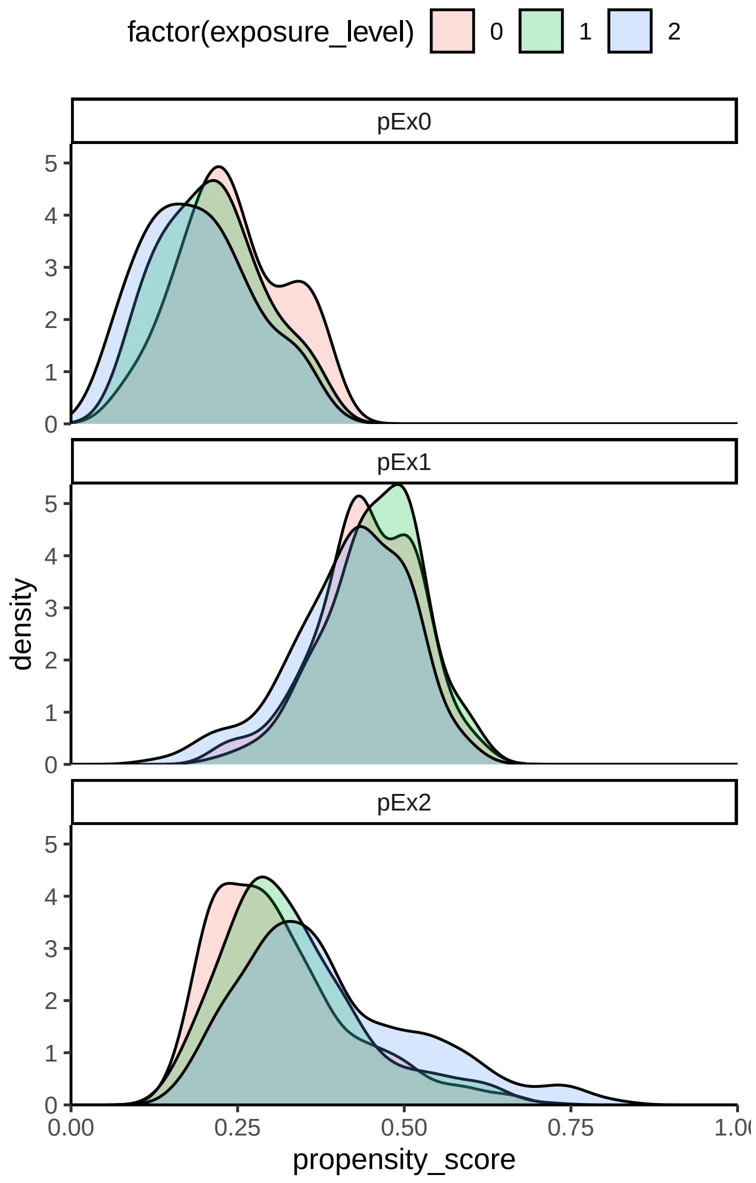


Figure 4: Propensity score overlap for exercise, a three level exposure in the NHEFS data.

```
## 6 0.23966182 0.4048806 0.3554575                 2
```

```
pscore_obs <- NULL
for(i in 1:nrow(ps_matrix)){
  pscore_obs <- rbind(pscore_obs,
                  ps_matrix[i, ps_matrix[i,]$Observed_Exercise + 1] # note the "+ 1"
                  )
}

head(pscore_obs)
```

```
##           [,1]
## [1,] 0.5409571
## [2,] 0.3245203
## [3,] 0.7784242
## [4,] 0.4192774
## [5,] 0.2867693
## [6,] 0.3554575
```

Each row in the `pscore_obs` object represents the predicted probability for each person being in their **observed exposure level**, conditional on the covariates in the propensity score model. With this, we can construct inverse probability weight as we did above. Note that we can stabilize these weights by including the overall mean of the observed exposure category. We can actually do this simply with an intercept only multinomial model, as we did above:

```
ps_model <- vglm(factor(exercise) ~ 1,
              data = nhefs,
              family = "multinomial")

ps_num <- cbind(predict(ps_model, type = "response"),  nhefs$exercise)

ps_num <- data.frame(ps_num)

names(ps_num) <- c("p0", "p1", "p2", "exercise_level")
```

```
head(ps_num)
```

```
##          p0        p1        p2 exercise_level
## 1 0.2104265 0.4407583 0.3488152              2
## 2 0.2104265 0.4407583 0.3488152              0
## 3 0.2104265 0.4407583 0.3488152              2
## 4 0.2104265 0.4407583 0.3488152              1
## 5 0.2104265 0.4407583 0.3488152              1
## 6 0.2104265 0.4407583 0.3488152              2
```

```
pscore_num <- NULL
for(i in 1:nrow(ps_num)){
  pscore_num <- rbind(pscore_num,
                      ps_num[i, ps_num[i,]$exercise_level + 1]
  )
}

nhefs$sw_exercise <- pscore_num/pscore_obs

summary(nhefs$sw_exercise)
```

```
##        V1
##  Min.   :0.4233
##  1st Qu.:0.8378
##  Median :0.9511
##  Mean   :0.9987
##  3rd Qu.:1.1272
##  Max.   :3.1733
```

With these weights we can now estimate the average treatment affect between exercise and our outcome of interest (in this case, `wt_delta`):

```
modelRD_ex <- lm(wt_delta ~ relevel(factor(exercise),
                                    ref = "2"),
                 weight = sw_exercise,
                 data = nhefs)

coeftest(modelRD_ex, .vcov = vcovHC)
```

```
##
## t test of coefficients:
##
##                                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            0.491784   0.026023 18.8980   <2e-16 ***
## relevel(factor(exercise), ref = "2")0 0.047758   0.042457  1.1249   0.2609
## relevel(factor(exercise), ref = "2")1 0.035561   0.034878  1.0196   0.3082
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A similar procedure can be used for continuous exposures. This procedure, referred to at the "quantile binning" method (Naimi et al., 2014), first categorizes the exposure into deciles or more, and then model this categorized exposure as above using a multinomial logistic regression model.

References

S. R. Cole and M. A. Hernán. Constructing inverse probability weights for marginal structural models. *Am J Epidemiol*, 168(6):656–64, 2008.

Masayuki Henmi and Shinto Eguchi. A paradox concerning nuisance parameters and projected estimating functions. *Biometrika*, 91(4):929–941, 12 2004.

Ashley Isaac Naimi, Erica E M Moodie, N Auger, and Jay S Kaufman. Constructing Inverse Probability Weights for Continuous Exposures: A Comparison of Methods. *Epidemiol*, 25(2):292–299, 2014.

J. M. Robins, S. D. Mark, and W. K. Newey. Estimating exposure effects by modelling the expectation of exposure conditional on confounders. *Biometrics*, 48(2):479–95, 1992.

Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.