Regression in Time-Fixed Settings: Propensity Scores

Ashley I Naimi

Spring 2023

Contents

Learning Objectives

- Define the propensity score and understand how it can be used to adjust for confounding.
- Understand when it might be preferable to use the propensity score versus an outcome model (e.g., marginal standardization) to estimate a treatment effect.
- Be able to identify different ways of using the propensity score to estimate treatment effects, including propensity score adjustment, stratification, matching, and inverse probability weighting.
- Understand how and why inverse probability of treatment weighting works.
- Be able to implement inverse probability weighting to quantify the average treatment effect on the risk difference, risk ratio, and odds ratio scales using R.
- Be able to use stabilized, stabilized normalized, as well as trimmed inverse probability weights.

1 Introduction to Propensity Score Methods

So far in this course, we've focused on obtaining quantitative estimates of the average treatment 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:

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.

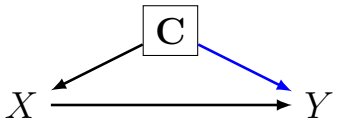


Figure 1: Directed acyclic graph depicting confounder adjustment using an outcome modelling approach. In this approach, information from the confounder to the outcome is 'blocked' (blue arrow). With this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome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

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: $1/[1 + exp(-1)]$

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

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

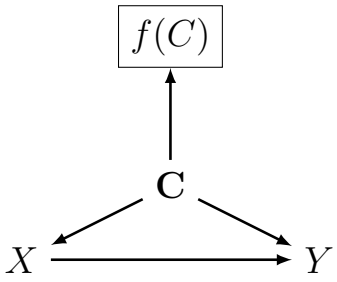


Figure 2: Directed acyclic graph depicting confounder adjustment using an outcome modelling approach. In this approach, information from the confounder to the outcome is 'blocked' (blue arrow). With this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome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

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 Adjustment, Matching, Stratification

To demonstrate these techniques, we will again use the NHEFS data:

```
#' Define where the data are
file_loc <- url("https://cdn1.sph.harvard.edu/wp-content/uploads/sites/1268/1268/20/nhefs.csv")

#' This begins the process of cleaning and formatting the data
nhefs <- read_csv(file_loc) %>%
    select(qsmk, wt82_71, wt82, wt71, exercise,
        sex, age, race, income, marital,
        school, asthma, bronch, starts_with("alcohol"),
        -alcoholpy, starts_with("price"),
        starts_with("tax"), starts_with("smoke"),
        smkintensity82_71) %>%
    mutate(income = as.numeric(income > 15),
        marital = as.numeric(marital > 2),
        alcoholfreq = as.numeric(alcoholfreq >
            1)) %>%
    na.omit(.)
```

```
factor_names <- c("exercise", "income", "marital",
    "sex", "race", "asthma", "bronch")
nhefs[, factor_names] <- lapply(nhefs[, factor_names],
    factor)

#' Define outcome
nhefs <- nhefs %>%
    mutate(id = row_number(), wt_delta = as.numeric(wt82_71 >
        median(wt82_71)), .before = qsmk)

#' Quick summary of data
nhefs %>%
    print(n = 5)
```

```
## # A tibble: 1,055 x 27
##      id wt_delta  qsmk wt82_71  wt82  wt71 exercise sex     age race  income
##   <int>    <dbl> <dbl>   <dbl> <dbl> <dbl> <fct>    <fct> <dbl> <fct> <fct>
## 1     1        0     0  -10.1   68.9  79.0 2        0        42 1     1
## 2     2        0     0    2.60  61.2  58.6 0        0        36 0     1
## 3     3        1     0    4.99  64.4  59.4 2        0        68 1     0
## 4     4        1     0    4.99  92.1  87.1 1        0        40 0     1
## 5     5        1     0    4.42 103.   99   1        1        43 1     0
## # ... with 1,050 more rows, and 16 more variables: marital <fct>, school <dbl>,
## #   asthma <fct>, bronch <fct>, alcoholfreq <dbl>, alcoholtype <dbl>,
## #   alcoholhowmuch <dbl>, price71 <dbl>, price82 <dbl>, price71_82 <dbl>,
## #   tax71 <dbl>, tax82 <dbl>, tax71_82 <dbl>, smokeintensity <dbl>,
## #   smokeyrs <dbl>, smkintensity82_71 <dbl>
```

The first step in any PS analysis is to obtain an estimate of the propensity score. This can be done using an array of techniques, including nonparametric techniques such as machine learning methods (e.g., Lee et al., 2010) or a more general nonparametric approach (Hirano et al., 2003) (see Technical Note), but is most often accomplished using logistic regression.

Technical Note:

Using the propensity score to adjust for confounding is a technique that falls into the class of single robust estimation. Singly robust estimators rely on a single model to adjust for confounding (or missing data or selection bias). In this case, the single model is the propensity score model.

Many researchers have proposed or implemented single robust estimation methods using machine learning methods. Machine learning methods consist of a wide range of analytic techniques that do not require hard to verify modeling assumptions. Because of this, they are often assumed to be less biased than their standard parametric counterparts. This perceived property has motivated many to either recommended or use machine learning methods to quantify exposure effects (Lee et al., 2010, Westreich et al. (2010),Snowden et al. (2011),Oulhote et al. (2019)). These "machine learning'' methods include techniques like kernel regression, splines, random forests, boosting, etc., which exploit smoothness across covariate patterns to estimate the regression function.

However, for any nonparametric approach there is an explicit bias-variance trade-off that arises in the choice of tuning parameters; less smoothing yields smaller bias but larger variance, while more smoothing yields smaller variance but larger bias (parametric models can be viewed as an extreme form of smoothing). This tradeoff has important consequences.

Convergence rates for nonparametric estimators become slower with more flexibility and more covariates. For example, a standard rate for estimating smooth regression functions is $N^{-\beta/(2\beta+d)}$, where β represents the number of derivatives of the true regression function, and d represents the dimension of, or number of covariates in, the true regression function. This issue is known as the **curse of dimensionality** (Györfi et al., 2002, Robins and Ritov (1997),Wasserman (2006)). Sometimes this is viewed as a disadvantage of nonparametric methods; however, it is just the cost of making weaker assumptions: if a parametric model is misspecified, it will converge very quickly to the wrong answer.

In addition to slower convergence rates, confidence intervals are harder to obtain. Specifically, even in the rare case where one can derive asymptotic distributions for nonparametric estimators, it is typically not possible to construct confidence intervals (even via the bootstrap) without impractically undersmoothing the regression function (i.e., overfitting the data) (Wasserman, 2006).

These complications (slow rates and lack of valid confidence intervals) are generally inherited by singly robust estimators such as methods that use only the propensity score for adjustment (this is apart from a few special cases which require simple estimators, such as kernel methods with strong smoothness assumptions and careful tuning parameter choices that are suboptimal for estimating f or g).

For general nonparametric estimators of the exposure or outcome model, convergence will be slow, and honest confidence intervals will not be computable (note that "honest" confidence intervals are defined as CIs with a minimum coverage probability no less than the nominal value over a rich class of nonparametric regression functions).

For these reasons, it is generally not advisable to use machine learning or nonparametric methods to estimate the propensity score and then proceed with PS adjustment in a regression model, matching, stratification, or weighting (Naimi et al., 2022). Instead, one should implement double robust estimation methods when machine learning or nonparametric methods are used.

In a previous lecture, we discussed using different link functions and distributions to estimate a conditionally adjusted risk difference, risk ratio, or odds ratio. However, when using parametric regression to estimate the propensity score, one would typically (always?)[2] use logistic regression:

[2] Generally, you will always want to use a method that bounds the predicted probabilities between 0 and 1.

```
# formulaVars <-
# paste(names(nhefs)[7:16],collapse =
# '+') modelForm <-
# as.formula(paste0('qsmk ~',
# formulaVars)) modelForm

# 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
```

2.1 Propensity Score Adjustment

For regression adjustment, we can then simply adjust for this propensity score in an outcome regression model to obtain an adjusted estimate of the parameter of interest. This is one of the easiest techniques available for using the propensity score to adjust for confounders:

```
# adjust for the propensity score to
# obtain conditionally adjusted risk
# difference
model1 <- glm(wt_delta ~ qsmk + propensity_score,
    data = nhefs, family = binomial("identity"))

summary(model1)$coefficients[2, ]
```

```
##     Estimate   Std. Error      z value     Pr(>|z|)
## 0.1458354801 0.0355240178 4.1052642424 0.0000403853
```

In the above output, the point estimate can be interpreted as the risk difference, with valid 95% CIs (binomial distribution, identity link). In principle, nothing here prevents us from using a logistic link function, and then marginalizing over the distribution the way we do with marginal standardization and the confounders (though we must use the bootstrap for standard error assessment).

2.2 Propensity Score Stratification

For propensity score stratification, we can create quantiles of the propensity score, and quantify the exposure effect within each stratum of this categorized propensity score. One can then combine the stratum specific point estimates and standard errors using a simple weighted average (Lunceford and Davidian, 2004). We can start by looking at the distribution of the propensity scores:

```
summary(nhefs$propensity_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05407 0.18392 0.24341 0.24739 0.30258 0.53682
```

We can create a variable that represents strata of the propensity score using quantiles. Often, *quintiles* of the distribution of the propensity score are used to create strata. However, it is important to note the bias-variance tradeoff that exists when selecting the number of quantiles of a distribution. A higher number of quantiles results in less bias and more variance, and vice versa. Here, we'll use quintiles to create strata:

```
## Identify the quintiles of the PS
quints <- quantile(nhefs$propensity_score,
    prob = seq(0, 1, by = 0.2), na.rm = T)

nhefs$ps_strata <- cut(nhefs$propensity_score,
    breaks = quints, include.lowest = T)

nhefs %>%
```

```
    group_by(ps_strata) %>%
    count()
```

```
## # A tibble: 5 x 2
## # Groups:   ps_strata [5]
##   ps_strata          n
##   <fct>          <int>
## 1 [0.0541,0.169]   211
## 2 (0.169,0.221]    211
## 3 (0.221,0.268]    211
## 4 (0.268,0.318]    211
## 5 (0.318,0.537]    211
```

Once the PS strata are created, the next step is to estimate the stratum specific associations of interest. Here, we quantify the risk differences using the binomial distribution and identity link:

```
# Perform logistic regression within
# each stratum

stratified_ps <- nhefs %>%
    group_by(ps_strata) %>%
    do(model = glm(wt_delta ~ qsmk, data = .,
        family = binomial("identity")))

summary(stratified_ps$model[[1]])$coefficients[2,
    ]
```

```
##   Estimate Std. Error    z value   Pr(>|z|)
## 0.06236559 0.10457758 0.59635716 0.55093665
```

```
summary(stratified_ps$model[[2]])$coefficients[2,
    ]
```

```
##   Estimate Std. Error    z value   Pr(>|z|)
## 0.15649264 0.08122227 1.92672092 0.05401442
```

```
summary(stratified_ps$model[[3]])$coefficients[2,
    ]
```

```
##   Estimate Std. Error    z value   Pr(>|z|)
## 0.23359684 0.07393250 3.15959605 0.00157988
```

```
summary(stratified_ps$model[[4]])$coefficients[2,
    ]
```

```
##    Estimate  Std. Error     z value    Pr(>|z|)
## 0.206470708 0.071961882 2.869167715 0.004115535
```

```
summary(stratified_ps$model[[5]])$coefficients[2,
    ]
```

```
##   Estimate Std. Error    z value   Pr(>|z|)
## 0.07534113 0.07087436 1.06302372 0.28777120
```

The stratum specific coefficients and standard errors can then be combined using standard equations:

$$\hat{\gamma} = \frac{1}{K}\sum_{k=1}^{K}\left\{\hat{\gamma}^{(i)}\right\}, \qquad SE(\hat{\gamma}) = \sqrt{\frac{1}{K^2}\sum_{k=1}^{K}\left\{SE(\hat{\gamma}^{(i)})^2\right\}}$$

These equations could be executed with, for example:

```
estimates <- NULL
for (i in 1:5) {
    estimates <- rbind(estimates, summary(stratified_ps$model[[i]])$coefficients[2,
        1])
}

mean(estimates)
```

```
## [1] 0.1468534
```

```
st.err <- NULL
for (i in 1:5) {
    st.err <- rbind(st.err, summary(stratified_ps$model[[i]])$coefficients[2,
        2])
}

sqrt((1/5^2) * sum(estimates^2))
```

```
## [1] 0.0724578
```

2.3 Propensity Score Matching

Finally, one can match on the basis of the propensity score, using a range of different techniques. These include 1:1 or 1:M matching, matching with or without replacement, greedy versus optimal matching, and nearest neighbor versus caliper matching (Austin, 2011).

```
library(MatchIt)
library(optmatch)
```

```
## The optmatch package has an academic license. Enter relaxinfo() for more information.
```

```
ps_matching <- matchit(formula = qsmk ~ exercise + sex + age + race + income + marital + school + asthma
                       method = "full", distance = "logit", # Distance defined by usual propensity scor
                       ratio = 1, # gives us 1:1 matching, which is the default
                       estimand = "ATE"
                       )

ps_matching
```

```
## A matchit object
##  - method: Optimal full matching
##  - distance: Propensity score
##              - estimated with logistic regression
##  - number of obs.: 1055 (original), 1055 (matched)
```

```
##  - target estimand: ATE
##  - covariates: exercise, sex, age, race, income, marital, school, asthma, bronch, alcoholfreq
```

Once we've identified the matched pairs, we can create a dataset that contains the information we need to match. The `matchit` function in `R`, as with some other matching approaches, constructs a set of weights to operationalize the matched sets (Yoshida et al., 2017). The dataset will include a weights variable and subclass variable that provide the information needed to conduct the matched analysis:

```
matched_data <- match.data(ps_matching)

matched_data %>%
    select(id, qsmk, exercise, sex, age,
        race, income, marital, school, asthma,
        bronch, alcoholfreq, propensity_score,
        distance, weights, subclass) %>%
    print(n = 6)
```

```
## # A tibble: 1,055 x 16
##      id  qsmk exercise sex     age race  income marital school asthma bronch
##   <int> <dbl> <fct>    <fct> <dbl> <fct> <fct>  <fct>    <dbl> <fct>  <fct>
## 1     1     0 2        0        42 1     1      0            7 0      0
## 2     2     0 0        0        36 0     1      0            9 0      0
## 3     3     0 2        0        68 1     0      1            5 0      0
## 4     4     0 1        0        40 0     1      0           11 0      0
## 5     5     0 1        1        43 1     0      1            9 0      0
## 6     6     0 2        0        51 0     1      0           10 0      0
## # ... with 1,049 more rows, and 5 more variables: alcoholfreq <dbl>,
## #   propensity_score <dbl>, distance <dbl>, weights <dbl>, subclass <fct>
```

One can then use this dataset with a standard `glm` model, weighted to implement the matching:

```
matched_model <- glm(wt_delta ~ qsmk, data = matched_data,
    weights = weights, family = binomial(link = "identity"))
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```
summary(matched_model)$coefficients
```

```
##              Estimate Std. Error   z value      Pr(>|z|)
## (Intercept) 0.4618364 0.01769257 26.103403 3.335496e-150
## qsmk        0.1506439 0.03496294  4.308674  1.642364e-05
```

This model creates a warning referring to "non-integer #successes in a binomial glm". This warning is referring to the fact that, while each individual contributes either 0 or 1 event to the likelihood function, with the weights added, these "contributions" become non-integer values. For example, if an individual's weight is 0.718, they will contribute a total of 0.718 events to the overall analysis. This is not actually a problem when we use weights to conduct an analysis. It's just the consequence of using weights. Thus, the only real problem is the warning that we get.

One way to avoid this warning[3] is to use the `quasibinomial` distribution instead of the `binomial` option. The coefficient from this model can be interpreted as an average treatment effect obtained via matching. The next step is to obtain a standard error for this parameter. This is where things get a little uncertain. There are generally two approaches used to obtain standard errors for a matching estimator: the robust (sandwich) variance estimator, and the bootstrap.

[3] This warning can sometimes create problems in the context of, e.g., a simulation study where the warning will prematurely interrupt the simulation.

Here is some code to implement the robust variance estimator. Note the need for the "subclass" argument and the ID argument, which allows us to account for the strata that we use to match the data:

```
library(lmtest)
library(sandwich)

coeftest(matched_model, vcov. = vcovCL, cluster = ~subclass +
    id)
```

```
##
## z test of coefficients:
##
```

```
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 0.461836   0.019388 23.8211 < 2.2e-16 ***
## qsmk        0.150644   0.040720  3.6995 0.0002161 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The problem here is that there is comparatively little evidence supporting the use of the robust (sandwich) variance estimator or the bootstrap for matching estimators. There is even some theoretical work suggesting that the bootstrap is biased for a propensity score matching estimator (Abadie and Imbens, 2008).

Thus, while matching is used regularly in the literature, there are some challenges in using propensity score matching estimators suggesting they may not be the best choice.

3 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:

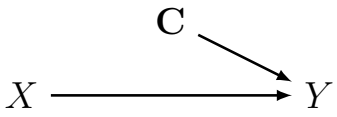


Figure 3: Directed acyclic graph depicting the causal relations between variables in a 'pseudo-population' obtained via inverse probability weighting. Again, with this adjustment, the exposure X os d-separated from the outcome Y, rendering the estimate of the exposure-outcome association unconfounded.

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:

```
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.

4 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.[4]

[4] 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[5]:

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

[5] 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:

```
# 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.
##  0.4758  0.8866  0.9702  0.9997  1.0698  3.1971
```

```
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> <fct> <dbl> <fct>                <dbl> <dbl>
## 1     1        0     0 0        42 2                    0.113 0.849
## 2     2        0     0 0        36 0                    0.148 0.883
## 3     3        1     0 0        68 2                    0.201 0.942
## 4     4        1     0 0        40 1                    0.326 1.12
## 5     5        1     0 1        43 1                    0.122 0.857
## # ... with 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.

```
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) 0.4637114 0.01770881 26.185357 8.430593e-117
## qsmk        0.1380809 0.03514616  3.928763  9.094879e-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:

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

summary(nhefs$sw_norm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1488  0.2773  0.3035  0.3127  0.3346  1.0000
```

```
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> <fct> <dbl> <fct>                <dbl> <dbl>   <dbl>
## 1     1        0     0 0        42 2                    0.113 0.849   0.266
## 2     2        0     0 0        36 0                    0.148 0.883   0.276
## 3     3        1     0 0        68 2                    0.201 0.942   0.295
## 4     4        1     0 0        40 1                    0.326 1.12    0.349
## 5     5        1     0 1        43 1                    0.122 0.857   0.268
## # ... with 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) 0.4637114 0.01770881 26.185357 8.430593e-117
## qsmk        0.1380809 0.03514616  3.928763  9.094879e-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%
## 1.824258
```

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

summary(nhefs$sw)
```

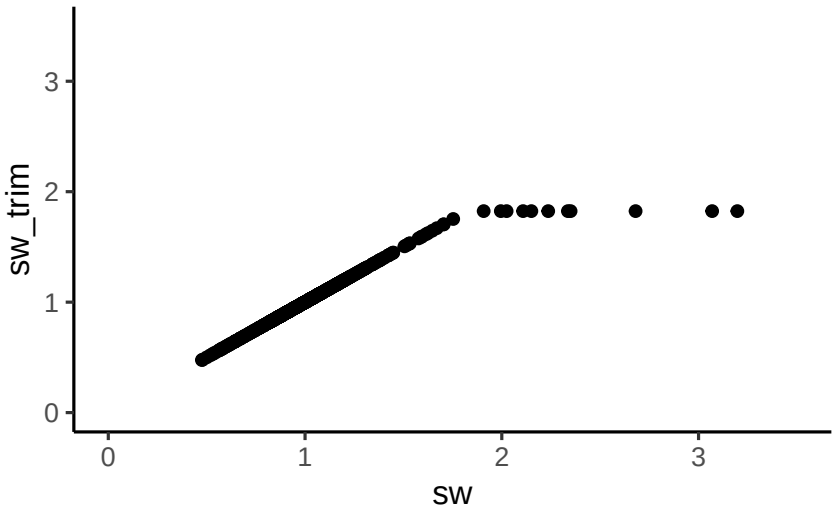
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4758  0.8866  0.9702  0.9997  1.0698  3.1971
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4758  0.8866  0.9702  0.9941  1.0698  1.8243
```

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

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



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:

```
library(lmtest)
```

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

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## (Intercept) 0.463711   0.017831 26.0055 < 2.2e-16 ***
## qsmk        0.138081   0.037492  3.6829 0.0002306 ***
## ---
## 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.

5 Takeaways

- The propensity score is defined as the probability of **being exposed** conditional on all confounders needed to attain exchangeability
- There are important practical distinctions between the true and estimated propensity score. Even when the true propensity score is available, it is often better to use the estimated propensity score when using it to estimate treatment effects.
- The propensity score is a **scalar summary** of all of the information contained in the confounding variables. For this reason, adjusting for the propensity score allows us to adjust for confounding.
- Propensity score adjustment can be implemented by fitting a propensity score model, and then fitting a model for the outcome of interest regressed against the exposure of interest, while conditioning on the estimated propensity score. When using this approach, model based standard errors for the exposure are valid.
- Propensity score stratification can be implemented by creating strata based on the propensity score. Quintiles are commonly used to construct strata, but there is a bias-variance tradeoff to consider when choosing the number

of strata. Stratum specific exposure outcome associations can then be averaged to obtain the overall exposure-outcome association of interest.

- Propensity score matching can be implemented in a number of different ways, including 1:M, nearest neighbors, caliper matching, optimal matching, greedy matching, etc. However, all of these approaches are affected by the absence of theoretically justified standard error estimators.
- The most common technique for using the propensity score in epidemiology is inverse probability weighting. IP weights can be used to balance the distribution of the confounders across exposure categories, thus removing the association between the exposure and the confounders, and thus adjusting for confounding.
- There are several variations of inverse probability of treatment weights, including stabilized, stabilized normalized, and trimmed inverse probability weights.
- Only the stabilized (or stabilized and trimmed) inverse probability weights should have a mean of 1 and a max that is not "too large"
- When using inverse probability weights, it's important to use either the robust (sandwich) variance estimator, or the bootstrap to obtain appropriate standard errors.

References

A Abadie and Guido W. Imbens. On the failure of the bootstrap for matching estimators. *Econometrica*, 76(6):1537–1557, 2008.

Peter C. Austin. Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies. *Pharmaceutical Statistics*, 10(2):150–161, 2022/03/22 2011. DOI: https://doi.org/10.1002/pst.433. URL https://doi.org/10.1002/pst.433.

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

L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, New York, NY, 2002.

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

Keisuke Hirano, Guido W. Imbens, and Geert Ridder. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71(4):1161–1189, 2003.

Brian K. Lee, Justin Lessler, and Elizabeth A. Stuart. Improving propensity score weighting using machine learning. *Stat Med*, 29(3):337–346, 2010.

Jared K. Lunceford and Marie Davidian. Stratification and weighting via the propensity score in estimation of causal treatment effects: a comparative study. *Statistics in Medicine*, 23(19):2937–2960, 2022/03/22 2004. DOI: https://doi.org/10.1002/sim.1903. URL https://doi.org/10.1002/sim.1903.

Al Naimi, A Mishler, and Edward H. Kennedy. Challenges in obtaining valid causal effect estimates with machine learning algorithms. *Am J Epidemiol*, kwab201, 2022.

Youssef Oulhote, Brent Coull, Marie-Abele Bind, Frodi Debes, Flemming Nielsen, Ibon Tamayo, Pal Weihe, and Philippe Grandjean. Joint and independent neurotoxic effects of early life exposures to a chemical mixture: A multipollutant approach combining ensemble learning and g-computation. *Environmental Epidemiology*, 3(5):e063, 2019.

J M Robins and Y Ritov. Toward a curse of dimensionality appropriate (coda) asymptotic theory for semi-parametric models. *Stat Med*, 16(1-3):285–319, Jan 1997.

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.

Jonathan M. Snowden, Sherri Rose, and Kathleen M. Mortimer. Implementation of g-computation on a simulated data set: Demonstration of a causal inference technique. *Am J Epidemiol*, 173(7):731–738, 2011.

Larry Wasserman. *All of nonparametric statistics*. Springer, New York; London, 2006.

Daniel Westreich, Justin Lessler, and Michele Jonsson Funk. Propensity score estimation: neural networks, support vector machines, decision trees (cart), and meta-classifiers as alternatives to logistic regression. *J Clin Epidemiol*, 63(8):826 – 833, 2010.

Kazuki Yoshida, Sonia Hernández-Díaz, Daniel H Solomon, John W Jackson, Joshua J Gagne, Robert J Glynn, and Jessica M Franklin. Matching weights to simultaneously compare three treatment groups: Comparison to three-way matching. *Epidemiology (Cambridge, Mass.)*, 28(3):387–395, 2017.