Principal Stratification for Intelligent Tutors: A Tutorial

Adam C Sales
University of Texas College of Education
asales@utexas.edu

John F Pane
RAND Corporation
jpane@rand.org

That some intelligent tutors help some students sometimes is now beyond doubt. That some intelligent tutors fail to help, or even hurt, some students sometimes, is also known. What is unknown is what drives this heterogeneity—why do tutors sometimes help, and sometimes hurt students' learning? Presumably, differences in how different students use educational technology plays an important role in how much that technology helps them learn. There is plenty of available data to measure student usage—primarily log data—and a growing number of randomized trials to assess tutors' effectiveness. However, estimating the relationship between usage and effectiveness requires surmounting subtle and difficult statistical challenges; standard methods are not available.

This paper describes and illustrates principal stratification, a causal inference framework that allows for causal modeling of log data from a randomized trial of an intelligent tutor. Using data from the Cognitive Tutor Algebra I effectiveness trial, it demonstrates two methods for conducting a principal stratification analysis, including code in R and Stan. It demonstrates the entire process of a principal stratification analysis, including data pre-processing, model formulation, fitting, and checking, and the examination and interpretation of results.

1. Introduction: Intelligent Tutors, Log Data, and Effectiveness

Between 2007 and 2009, a team of researchers at the RAND Corporation conducted a multi-state randomized trial of the Cognitive Tutor Algebra I (CTAI) curriculum. Algebra I classrooms in schools randomly selected for intervention were granted access to the Cognitive Tutor software, an intelligent tutoring system, along with a student-centered curriculum and textbook, both produced by the Carnegie Learning corporation. Schools selected for the control condition continued with business as usual.

The results (Pane et al., 2014) were mixed—a negative, statistically insignificant effect on post-test scores in the first year and a substantial positive effect, statistically significant in the high school sample, in year 2. Evidently, CTAI effected different students differently. What drove these differences? Some of the differences may be explained by students' characteristics—for example, it may have had a different effect for high and low performing students, or for students of different socioeconomic backgrounds. Relatively straightforward subgroup analyses or treatment moderation methods are available to study these differences. Another likely driver of treatment effect heterogeneity was usage—students and teachers who used the CTAI curriculum differently benefited (or, possibly, were hurt) to different extents.

The RAND experiment produced a large and rich auxiliary dataset to study usage of the software component of CTAI—computerized log data. Over the course of the study, Carnegie Learning recorded which students worked which portions of the software on what dates, for how long, and with what results. How does software usage correspond with treatment effects? Are some styles of usage more effective than others?

These types of data, and these types of questions, are not unique to CTAI. The past decade has seen a number of large randomized field trials of educational technology (Escueta et al., 2017). Each of these presumably generated log data, that can now be used to answer (and spur) questions about the relationship between software usage and treatment effects.

This paper is a tutorial for methods to answer these types of questions under the framework of principal stratification (PS) (Angrist et al., 1996; Frangakis and Rubin, 2002; Page, 2012). PS aims to estimate treatment effects for subsets of a sample defined by "intermediate variables"—variables that may, themselves, have been affected by treatment assignment. Software usage is one example—assignment to treatment gave students access to the software—but other examples abound. For instance, attrition or missing outcome data, compliance with treatment assignment, and treatment-effect mediators are all intermediate variables. In this paper, we will describe PS with the goal of helping intelligent tutoring researchers apply the methods to their own studies. We will describe the PS framework—its targets of estimation and their rationale—and two methods for estimating PS models. Along the way, we will demonstrate these methods with a simplified analysis of the RAND CTAI dataset, including code in `R` (R Core Team, 2016) and Stan (Stan Development Team, 2016). Recent work, by ourselves and others, has shown that estimating PS models may be harder than previously thought—for that reason, we will devote considerable space to model checking and validation.

2. Running Example: Starting at the Beginning

To demonstrate the ideas and practice of PS for ITS experiments, we will focus on a particular example. We present the analyses to illustrate the method, not to evaluate CTAI.

The Cognitive Tutor is built on the theory of mastery learning—students progress through a set curriculum at their own pace, as they master each of its skills. By default, students in the CTAI curriculum were supposed to begin with the first unit, "Linear Patterns." However, a number of students in the intervention group in the second year of the study began their Algebra I work with a later section. While it is hard to know the true reasons some students began with a later section, it may be because their teachers believed that they had already mastered "Linear Patterns" and could safely skip it. In that case, we might imagine that those students experienced greater treatment effects by avoiding wasting time on sections they had already mastered. On the other hand, they may experience smaller treatment effects if their teachers' judgment was mistaken, or if working earlier sections helps in other ways. To distinguish between these hypotheses, it may be useful to estimate the CTAI treatment effect for two groups of students: "firsters," who started at the first unit, and "skippers," who started at later units. Of course, skippers and firsters likely differ in more than just the first units they worked—interpreting any difference between treatment effects for the two groups will take some care.

The following code in R loads the dataset we will be using, which is available for download at http://tiny.cc/psEDMdata.

```r
dat <- read.csv('CTAIdata.csv')
```

Here is a short summary of its contents:

	Treatment	Control
Y		
	0.06 (0.86)	-0.34 (0.75)
begin		
firster	66%	
skipper	34%	
grade		
9	87%	94%
higher	13%	6%
race		
HispAIAN	59%	57%
BlackMulti	33%	40%
WhiteAsian	8%	3%
sex		
F	53%	48%
M	47%	52%
frl		
	0.52 (0.5)	0.55 (0.5)
spec		
typical	78%	73%
speced	13%	19%
gifted	9%	9%
esl		
	0.07 (0.25)	0.14 (0.35)
pretest		
	-0.07 (0.89)	-0.16 (0.81)
n	654	661

The dataset contains outcomes Y—Algebra I posttest scores—begin, an indicator for whether treated students are firsters or skippers (it is NA for control students), and a set of baseline covariates: grade takes values of 9 or higher, for students in 9th or higher grades, respectively, race, with combined categories of Hispanic/American Indian/Alaskan Native, Black/Multiracial, and White/Asian, sex, frl, an indicator for students who receive free or reduced-price lunch, spec, which categorizes students as typical, gifted, or receiving special education, `esl`, an indicator for students learning English as a second language, and `pretest`, an Algebra I pretest comparable to the post-test but take before the onset of the experiment. The last line of the table, labeled `n`, gives the sample sizes in the two groups: 654 treated students and 661 controls.

This dataset contains a subset of the students and of the variables included in the larger CTAI dataset. Specifically, it includes only data from the second year of the study, collected from intervention schools in Texas in which usage data was observed, along with their matched controls. It also excludes nesting information—the intervention was assigned at the school level, and within schools, students were nested within teachers and classrooms. Finally, it elides missing data issues: missing covariate data were imputed in advance and intervention students with missing usage data were discarded.

These simplifications were meant to ease exposition—as we shall see, estimating PS models is a rather complex process, even without clustered and missing data. For that reason, the results we present here should not be taken seriously; the point is to demonstrate PS modeling, not to answer substantive questions about the CTAI effect.

3. Causal Inference and Principal Stratification

3.1. Average Effects from Randomized Experiments

In the Neyman-Rubin Causal Model (Holland, 1986; Rubin, 1978), causation is, in essences, a contrast between potential outcomes. If, say, Y_i is the posttest score for student i, then Y_{Ti} is the posttest score i would exhibit *were i assigned to the treatment condition* and Y_{Ci} is i's score if i is assigned to the control condition. Crucially, the potential outcomes Y_T and Y_C are defined based on treatment *assignment*, not actual receipt of treatment. This follows the "intent to treat" principle (Lachin, 2000). Since students can only be assigned to one of the two conditions, for each student i only one of Y_{Ti} and Y_{Ci} is observed—researchers observe Y_{Ti} for students assigned to treatment and Y_{Ci} for those assigned to control. The other, missing potential outcome needs to be estimated. Since reliably estimating individual values Y_{Ci} and Y_{Ti} is often impossible, the goal of causal inference is typically to estimate something like the average treatment effect (ATE):

$$ATE = \mathbb{E}[Y_T - Y_C] = \mathbb{E}[Y_T] - \mathbb{E}[Y_C] \tag{1}$$

the average difference between how students would score if assigned to treatment and how they would score under control. Importantly, the expectations $\mathbb{E}[Y_T]$ and $\mathbb{E}[Y_C]$ are taken over the *entire* sample, not just for students assigned to the treatment or to the control condition. This is possible because Y_{Ti} and Y_{Ci} are both defined for every study participant, even if only one is observed. Randomization of treatment assignment allows us to estimate average potential outcome values for the entire sample, $\mathbb{E}[Y_T]$ and $\mathbb{E}[Y_C]$, with the sample averages of Y for the treatment group and control group, respectively.

Subgroup analysis is a straightforward extension, as long as the definition of the subgroup is not itself affected by treatment assignment. Take gender, for example: the ATE for women is the difference between $\mathbb{E}[Y_T|Female]$, the mean Y_T for all of the women in the study, and

$\mathbb{E}[Y_C|Female]$, the mean Y_C for the women in the study. Along exactly the same lines, the ATE for men is $\mathbb{E}[Y_T - Y_C|Male]$. Each of these is a causal effect and may be estimated by differences between sample averages under randomization. On the other hand, the difference between the two effects, the ATE for women and the ATE for men, is not a causal comparison. Technically, this is because nowhere do we model potential outcomes that are a function of gender—there is no Y_{Mi} or Y_{Fi} representing i's outcome were i male or female, respectively. More prosaically, gender was not randomized, so nothing guarantees that the men and women of the study are causally comparable.

3.2. Intermediate Variables and Principal Stratification

When subgroups are defined subsequent to treatment, the logic of subgroup analysis ceases to hold. Take the skipper/firster example described above—recall, firsters begin Algebra I with the first section, while skippers begin with a later section. The average outcome among treated firsters estimates $\mathbb{E}[Y_T|Firster]$ as in the gender case. However, how are we to make sense of $\mathbb{E}[Y_C|Firster]$? Since subjects in the control group do not have access to the software, they are neither firsters nor skippers. Is Y_C a meaningful quantity for skippers? Is the skipper/firster distinction meaningful for members of the control group? Along similar lines, how might the ATE for firsters be estimated? Comparing firsters' outcomes to the entire control group is not a fair comparison, since firsters may differ, on average, from the rest of the sample, in some relevant baseline characteristic.

In other examples, the intermediate value is defined for both treatment and control groups. For instance, if members of the control group had been able to access the CT software, there may have been control firsters and skippers. Nevertheless, the problem remains—skipping the first section in the control condition may not be quite the same as skipping it in the treatment condition. Further, it may be the case that those students who skipped the first section in the control condition may have worked the first section in the treatment condition, and vice-versa. Comparing control firsters to treatment firsters risks comparing groups that are not really comparable, and undermines the rationale of randomization.

The PS solution to this conundrum is to consider *potential* values of intermediate variables. Let M_i be a categorical variable that encodes whether subject i is a firster or a skipper. Then M_{Ti} encodes the group i would be in were i assigned to the treatment condition. Unlike M, M_T is defined for both treated and untreated subjects—$M_{Ti} = Firster$ (or $M_{Ti} = F$ for short) means that if i had been assigned to treatment, i would have been a firster. Whereas $\mathbb{E}[Y_C|M = Firster]$ may be undefined, is the mean outcome under the control condition for subjects $\mathbb{E}[Y_C|M_T = Firster]$ who *would be* firsters if assigned to treatment. The variable M_T defines two *principal strata*: students for whom $M_T = Firster$ and those for whom $M_T = Skipper$. Assuming that no one assigned to treatment fails to even begin using CT, every subject in the experiment, regardless of treatment assignment, is a member of one of the two strata. However, principal stratum membership M_T is only observed for students assigned to treatment—it is unknown which control students are potential firsters and which are potential skippers. The ATE $\mathbb{E}[Y_T - Y_C|M_T = Firster]$, called a *principal effect*, is a straightforward subgroup effect, no different than the ATE for women or for men. However, estimating principal effects is much harder than typical subgroup effects, since M_T is unobserved in the control group.

When M is well defined for both treatment groups, (for instance, if members of the control group had access to CT software), M_{Ci} encodes subject i's grouping were i assigned to control. In such scenarios, principal strata are defined based on both M_T and . In the CT example, there would be four principal strata: students who would be firsters in both treatment conditions, $M_{Ti} = M_{Ci} = Firster$, students who would be skippers in both conditions, $M_{Ti} = M_{Ci} = Skipper$, students who would be skippers in the control condition and firsters under treatment (i.e. whom the treatment causes to be firsters), $M_{Ti} = Firster$, $M_{Ci} = Skipper$, and those who would be firsters under control but skippers under treatment, $M_{Ti} = Skipper$, $M_{Ci} = Firster$. This is the more typical PS setup (e.g. (Angrist et al., 1996; Feller et al., 2016; Page, 2012)). However, we will focus on case in which only M_T is well defined, since this case is both simpler and more relevant to randomized trials of educational technology.

4. Non-Parametric Bounding for Principal Effects

Even with infinite data, principal effects are unknowable, absent additional assumptions. There is simply know way to know for sure which control subjects are firsters and which are skippers. However, even without imposing a parametric model, the data imply bounds on principal effects—limits on how small or large they may be. Miratrix et al. (2017) provides a nice overview of this method, describing its use in a complex PS design, and discussing the use of covariates to tighten the bounds. While we will draw from that paper the approach we demonstrate here is more similar to what was described in Lee (2009).

Say our goal is to estimate the ATE for potential firsters, $\mathbb{E}[Y_T|M_T = F] - \mathbb{E}[Y_C|M_T = F]$. The first term, $\mu_{TF} \equiv \mathbb{E}[Y_T|M_T = F]$, may be estimated by the average outcomes among treated firsters, $\hat{\mu}_{TF} = \bar{Y}_{Z=0;M=F}$. The goal here will be to put bounds on the second term, $\mu_{CF} \equiv \mathbb{E}[Y_C|M = Firster]$.

The two crucial pieces of data are the Y values observed in the control group and the overall proportion of the sample comprised by firsters, $\pi_F = Pr(M_T = F)$, estimated by the proportion of the treated sample with $S = F$, $\hat{\pi}_F = \sum_{Z=1}[S = Firster]/n_T$, where $[\cdot]$ is the indicator function, equal to one when its contents are true and 0 otherwise, and n_T is the size of the treatment group. In our dataset,

```r
pi_F <- with(dat,mean(begin[treatment==1]=='firster'))
pi_F

## [1] 0.656
```

Because of randomization, the proportion of the treatment group with $M_T = F$ estimates the same proportion in the entire experimental sample.

For the time being, assume that π_F is estimated perfectly, so that $\hat{\pi}_F = \pi_F$. Then the number of potential firsters in the control group is

$$n_{CF} = \lfloor \pi_F * n_C \rfloor .$$

(For the sake of bounding we truncate n_{CF} with the floor function $\lfloor \cdot \rfloor$.)

```
n_C <- sum(dat$treatment==0)
n_cf <- trunc(pi_F*n_C)
n_cf

## [1] 433
```

Then a lower bound for μ_{CF} is the mean of the n_{CF} lowest control Y values, and an upper bound is the mean of the highest n_{CF} control Ys. Formally, let

$$Y_{(1)}^{Z=0} \leq Y_{(2)}^{Z=0} \leq \dots Y_{(n_C)}^{Z=0}$$

be the sorted Ys in the control group. Then the bounds are

$$\mu_{CF}^{L} = \frac{1}{n_{CF}} \sum_{i=1}^{n_{CF}} Y_{(i)}^{Z=0} \text{ and } \mu_{CF}^{U} = \frac{1}{n_{CF}} \sum_{i=n_C-n_{CF}+1}^{n_C} Y_{(i)}^{Z=0}$$

which may be calculated in R as:

```
YcSort <- with(dat,sort(Y[treatment==0]))
mu_cfL <- mean(YcSort[1:n_cf])
mu_cfU <- mean(YcSort[(n_C-n_cf+1):n_C])
```

Finally, ATE for firsters is $\mu_{TF} - \mu_{CF}$ can be bounded as:

$$\mu_{TF} - \mu_{CF}^{U} \leq \mu_{TF} - \mu_{CF} \leq \mu_{TF} - \mu_{CF}^{L}$$

In our data,

```
mu_tf <- with(dat,mean(Y[treatment==1]))
c(mu_tf-mu_cfU,mu_tf-mu_cfL)

## [1] -0.01162  0.84292
```

Unfortunately, this interval is rather wide. That said, it suggests that CTAI assignment could not have lowered firster's test scores by much, and may have increased them substantially.

4.1. Incorporating Uncertainty into Bounds

Estimating bounds on principal effects with uncertainty is an area of open research. Miratrix et al. (2017) suggests using the bootstrap (Efron et al., 1979). Although the theoretical rationale for the bootstrap in this case is not yet well established, it performs well in simulation studies.

First, it will be convenient to wrap the above bounding steps into a function in R :

```r
bounds <- function(dat){
    pi_F <- with(dat,mean(begin[treatment==1]=='firster'))
    n_C <- sum(dat$treatment==0)
    n_cf <- trunc(pi_F*n_C)
    YcSort <- with(dat,sort(Y[treatment==0]))
    mu_cfL <- mean(YcSort[1:n_cf])
    mu_cfU <- mean(YcSort[(n_C-n_cf+1):n_C])
    mu_tf <- with(dat,mean(Y[treatment==1]))
    c(lower=mu_tf-mu_cfU,upper=mu_tf-mu_cfL)
}
```

The bootstrap is a technique to mimic the process of drawing a random sample from a population, and may be used to estimate standard errors and confidence intervals for a very wide range of statistics of arbitrary complexity. A large number (say B) of times, draw a "bootstrap sample" from the data, by randomly sampling n cases *with replacement*. Then, for each bootstrap sample, calculate a new set of bounds:

```r
B <- 1000
bsBounds <- matrix(nrow=B,ncol=2)
for(b in 1:B){
    bsSamp <- sample(1:nrow(dat),nrow(dat),replace=TRUE)
    bsBounds[b,] <- bounds(dat[bsSamp,])
}
```

Finally, a the 5th percentile of the lower bound and 95% percentile of the upper bound form a 95% confidence interval for the ATE for firsters.

```r
c(quantile(bsBounds[1,],0.05), quantile(bsBounds[2,],0.95))

##       5%      95%
## 0.04076 0.87624
```

Miratrix et al. (2017) includes a brief discussion of theoretical issues with the bootstrap for PS bounds (with an explanation of why the relevant percentiles are 5 and 95, and not the usual 2.5 and 97.5) along with simulation results.

Clustered or stratified randomization, or other complex randomization schemes, require modification of the bounding procedure; Some possibilities are discussed in Miratrix et al. (2017).

5. Model Based Principal Stratification in Theory

Non-parametric bounds for principal effects are guaranteed to be correct, but are typically quite wide—often too wide to be of scientific or practical use. In contrast, assuming models for potential outcomes Y_T and Y_T as a function of principal stratum and covariates, and stratum M_T as a function of covariates, can sometimes yield precise point estimates for principal effects. These models jointly predict which control subjects belong to which stratum, helping to solve the fundamental problem of principal stratification—that M_T is unobserved in the control group. Additionally, as we shall see, models for potential outcomes are necessary to correct for uncertainty in M_T imputations. Of course, if the model is (sufficiently) misspecified, or if problems arise in the fitting process, these estimates may be wrong. Extensive model checking even more important than usual in this case.

It is instructive to begin with the case in which there are no covariates. Say the researcher models potential outcomes Y_T and Y_C as normal, but allows their means and standard deviations to vary with both treatment assignment and M_T. Then there are eight parameters: four means denoted with μ: μ_{TF} and μ_{CF} for treatment and control firsters, and μ_{TS} and μ_{CS} for treatment and control skippers, and four standard deviations denoted with σ: σ_{TF}, σ_{CF}, σ_{TS}, and σ_{CS}. The average treatment effect of being assigned to treatment is $\mu_{TF} - \mu_{CF}$ for firsters and $\mu_{TS} - \mu_{CS}$ for skippers.

As in the bounding case, estimating μ_{TF}, μ_{TS}, σ_{TF}, and σ_{TS}, the parameters for the treatment group, is straightforward. However, since control students could be either firsters or skippers—M_{Ti} is unobserved—estimating μ_{CF} and μ_{CS} is more difficult. On the other hand, randomization ensures that probability of being a firster, $\pi \equiv Pr(M_T = F)$, is the same in the treatment and control groups, and can therefore be estimated. If π is the probability of being a firster, then $1 - \pi$ is the probability of being a skipper.

Since we are not sure whether $M_{Ti} = F$ or $M_{Ti} = S$, we do not know which distribution i's outcome Y is drawn from. Instead, it is drawn from a mixture model:

$$Y_C \sim \pi \mathcal{N}(\mu_{CF}, \sigma_{CF}) + (1 - \pi)\mathcal{N}(\mu_{CS}, \sigma_{CS}) \tag{2}$$

In other words, with probability π, Y_C is drawn from the distribution for control firsters, and with probability $1 - \pi$, Y_C is drawn from the distribution for skippers.

The role of covariates are to personalize the probabilities π and the means μ (and, in principal, standard deviations σ, but this is rarely done in practice) in the mixture model (2). Denote a vector of p covariates for student i as $\boldsymbol{x}_i = \{x_{1i}, \dots, x_{pi}\}$. Then, rather than use the same probability π for every control subject, as in (2) we may define individual probabilities, $\pi_i \equiv Pr(M_{Ti} = F) = Pr(M_T = F|\boldsymbol{x}_i)$. These may be estimated in the treatment group by a "usage model," such as logistic regression:

$$logit(\pi_i) = \boldsymbol{x}_i^t \boldsymbol{\beta}^U = \beta_0^U + \beta_1^U x_{1i} + \beta_2^U x_{2i} + \cdots + \beta_p^U x_{pi} \tag{3}$$

where $logit(x) = log\{x/(1-x)\}$. Because of randomization, this model can be extrapolated from the treatment group estimating probabilities $logit(\hat{\pi}_i) = \boldsymbol{x}_i^t \hat{\boldsymbol{\beta}}^U$ for i in the control group. (That said, if the number of covariates p is large compared with the sample size n, overfitting in the treatment group can cause problems, which may, in turn, be ameliorated with weakly-informative priors (Gelman et al., 2008).)

Covariates can also be used to predict potential outcomes, in an "outcome model." When outcomes are continuous, this is typically a linear regression, for instance

$$Y = \alpha_{ZM} + \beta_1^Y x_1 + \cdots + \beta_p^Y x_p + \epsilon \tag{4}$$

where ϵ is a random regression error, typically normally distributed, perhaps with a standard deviation that depends on Z and/or M_T. In model (4), the differences between firsters and skippers in treatment and control appears in the intercept α_{MZ} which takes four values: α_{TF} for treated firsters, α_{TS} for treated skippers, α_{CF} for control firsters and α_{CS} for control skippers. Average treatment effects can be expressed as differences between the intercepts: the mean of $Y_T - Y_C$ is $\alpha_{TF} - \alpha_{CF}$ for firsters and $\alpha_{TS} - \alpha_{CS}$ for skippers. We have found that including the same set of covariates in both the usage and outcome models can help correct for mild model misspecification.

Of course, since M_T is unobserved for the control group, (4) cannot be fit in the usual way for control students. Instead, control potential outcomes follow a mixture distribution, not unlike (2):

$$Y_{Ci} \sim \pi_i \mathcal{N}(\alpha_{CF} + \boldsymbol{x}_i^t \boldsymbol{\beta}^Y, \sigma_{CF}) + (1 - \pi_i)\mathcal{N}(\alpha_{CS} + \boldsymbol{x}_i^t \boldsymbol{\beta}^Y, \sigma_{CS}) \tag{5}$$

where the vector dot product $\boldsymbol{x}_i^t \boldsymbol{\beta}^Y = \beta_1^Y x_{1i} + \cdots + \beta_p^Y x_{pi}$. Now the normal means have been replaced with the linear model from 4 and the probabilities π have been replaced with personalized output from the usage model π_i.

Model (4) assumes that Y_C and Y_T behave quite similarly—the form of the model, as well as the covariate slopes β^Y are assumed to be the same regardless of treatment assignment or principal stratum. To a certain extent, this can be relaxed, and an analyst can estimate a different model for Y_C and Y_T. Indeed, since Y_C is not observed for treated subjects, and Y_T is not observed for controls, there is no reason, in principle, that the two should follow the same model. This is in contrast to the usage model (3), which can be extrapolated from one group to the other. The quantities M_T and $\boldsymbol{x}$ are independent of treatment assignment, whereas Y is not. In fact, unlike the distribution of Y_T conditional on $\boldsymbol{x}$ and M_T, the model for Y_C conditional on M_T cannot be estimated directly from the data without a model. The model for Y_C given $\boldsymbol{x}$ and M_T is an unverifiable assumption (though it is, of course constrained—its predictions must track observed Y_C values).

5.1. Fitting the Model

These ingredients—the outcome models for treatment and control students, and the usage model for treatment students—together define a "likelihood," a probability distribution for all of the observed data. We denote the likelihood as $L(\boldsymbol{\theta}_Y, \boldsymbol{\theta}_M; \boldsymbol{Y}, \boldsymbol{M}, \boldsymbol{x})$ a function of two sets of parameters—the parameters of the outcome model, $\boldsymbol{\theta}_Y$, and for the usage model $\boldsymbol{\theta}_M$—and of the data, outcomes $\boldsymbol{Y}$, usage $\boldsymbol{M}$, and covariates $\boldsymbol{x}$. When students are mutually independent, the likelihood is a product of the distributions of each student's data. In practice, we work with the log-likelihood $l(\boldsymbol{\theta}_Y, \boldsymbol{\theta}_M; \boldsymbol{Y}_T, \boldsymbol{Y}_C, \boldsymbol{M}, \boldsymbol{x})$, taking the natural logarithm of the likelihood; then this product becomes a sum. With the usage and outcome models defined in (3), (4), and (5), this comes out to:

$$\sum_{\substack{\text{Treated}\\\text{Students}}} log\{\text{Bernoulli}(M_i|logit^{-1}(\boldsymbol{x}_i^t\boldsymbol{\beta}^U))\} \tag{6}$$

$$+\sum_{\substack{\text{Treated}\\\text{Students}}} log\{\text{Normal}(Y_i|\alpha_{TM_i}+\boldsymbol{x}_i^t\boldsymbol{\beta}^Y,\sigma_{TM_i})\} \tag{7}$$

$$\begin{aligned}+\sum_{\substack{\text{Control}\\\text{Students}}} log\{&\pi_i\text{Normal}(Y_i|\alpha_{CF}+\boldsymbol{x}_i^t\boldsymbol{\beta}^Y,\sigma_{CF})\\ &+(1-\pi_i)\text{Normal}(Y_i|\alpha_{CS}+\boldsymbol{x}_i^t\boldsymbol{\beta}^Y,\sigma_{CS})\}\end{aligned} \tag{8}$$

Where Bernoulli$(m|\pi)$ is the probability of a Bernoulli trial with probability π yielding outcome m, i.e. π if $m=1$ and $1-\pi$ if $m=0$, and Normal$(y|\mu,\sigma)$ is the probability density function of a normal random variable with mean μ and standard deviation σ evaluated at y.

Classical maximum likelihood estimates for the model parameters are the values that maximize the log likelihood (6)–(8). To fit the model in a Bayesian framework requires a prior distribution for each of the model parameters. These are distributions that do not depend on the data—they may encode the researcher's prior beliefs or hunches about model parameters or serve as tools to regularize model fitting. Often, "uninformative" or "weakly informative" priors are available when researchers have little prior information about model parameter values; in fact, these are the default in Stan, the Bayesian software we will use. However, in some cases information about what values are more plausible than others (for instance, that effect sizes are probably not larger than 2) can go a long way in improving a model's fit. There is a large literature surrounding the formulation of Bayesian models and priors; for instance, Kadane (2011) or Gelman and Shalizi (2013).

The goal of a Bayesian analysis is a posterior distribution, the joint distribution of model parameters conditional on the data. The posterior distribution encodes information such as point estimates (typically taken as the distribution's modes, medians, or means) and uncertainty about the parameters' values, which can be expressed as probabilities. For instance, a 95% credible interval for a paramter θ is an interval $[a,b]$ such that $Pr(a\leq\theta\leq b)=0.95$. For a more complete overview of applied Bayesian statistics, see Gelman et al. (2014) or Kruschke (2014).

6. Model Based Principal Stratification in Practice

In this section, we will guide the reader through a Bayesian model based principal stratification analysis step-by-step. We will use `R` and Stan to fit the Bayesian model. `R` may be downloaded at `https://cran.r-project.org/`; A number of tutorials are available on the internet; see, for instance, Paradis (2002). Instructions for installing and using Stan are available at `http://mc-stan.org/`.

6.1. Writing a PS Model in Stan

Stan is stand-alone program that can be called from `R` . It uses Markov Chain Monte Carlo techniques to fit Bayesian models; for an overview, see Gelman et al. (2015). For more details see the

Users' Guide at `http://mc-stan.org`.

Typically, we write code for the Stan model in a separate document. In this paper, we will do so by defining character strings in `R` for each part of the model, then joining them and exporting them to a file, `psMod.stan`.

The syntax is divided into at least three parts (ours will have five). The first enumerates the data used to fit the model. Rather than a data frame as in `R`, it is a list of variables, of varying types. In the next section, we will create each of these variables in `R` to feed to stan. Note that the double forward slash `//` delimits comments.

```
dataBlock <- '
data{
 int<lower=1> nc; // number of control subjects
 int<lower=1> nt; // number of treated subjects
 int<lower=0> ncov; // number of covariates

 real Yctl[nc]; // control outcomes
 real Ytrt[nt]; // treatment outcomes

 matrix[nc,ncov] Xctl; // covariate matrix for controls
 matrix[nt,ncov] Xtrt; //covariate matrix for treated

 int<lower=0,upper=1> first[nt]; // 1=firster 0=skipper
}
'
```

We will feed Stan separate outcome vectors and covariate matrices for the treatment and control groups, since they play different roles in the model.

The next section lists parameters—these are the aspects of the model that we will want to inspect after it has been fit:

```
paramBlock <- '
parameters{
 real alphaTF; // Y-intercept for treatment firsters
 real alphaTS; // Y-intercept for treatment skippers
 real alphaCF; // Y-intercept for control firsters
 real alphaCS; // Y-intercept for control skippers

 real alphaU; // intercept for usage model
 vector[ncov] betaU; // coefficients for covariates in usage model
 vector[ncov] betaY; // coefficients for covariates in outcome model

 // residual standard deviation:
 real<lower=0> sigTF;
 real<lower=0> sigTS;
 real<lower=0> sigCF;
 real<lower=0> sigCS;
}
'
```

We will include an optional section called "transformed parameters," which is useful for inspecting functions of the declared parameters. Each of these must first be declared, along with a variable type, and then defined as a function of the parameters in the `parameters` section.

```
tranParamBlock <- '
transformed parameters{
 //declare new parameters:
 real skipperATE; //Avg. Effect for skippers
 real firsterATE; //Avg. Effect for firsters
 real ATEdiff; //Difference btw Avg Effects
 vector[nc] piC; //Pr(Firster) for controls
 vector[nt] piT; //Pr(Firster) for treateds

 //define new parameters:
 skipperATE=alphaTS-alphaCS;
 firsterATE=alphaTF-alphaCF;
 ATEdiff=firsterATE-skipperATE;
 piC=inv_logit(alphaU+Xctl*betaU);
 piT=inv_logit(alphaU+Xtrt*betaU);
}
'
```

The `model` section defines the statistical model—prior distributions and the likelihood (6)–(8).

We will introduce additional parameters into this section that are functions of the parameters declared earlier. The only difference between these and the parameters in `transformed parameters` is that these are not saved and reported with the model fit. Stan is most efficient when its code is "vectorized," i.e. written in the form of vectors and matrices instead of individual data-points. Some of this code will facilitate vectorization.

```
modelBlock1 <- '
model{
 // vectors of intercepts and residual SDs for treated
 // students. useful for vectorizing:
 vector[nt] alphaT;
 vector[nt] sigT;

 // The ? functions as if-else A?B:C returns B if A=1 and C otherwise
 for(i in 1:nt){
  alphaT[i]= first[i]?alphaTF:alphaTS;
  sigT[i]= first[i]?sigTF:sigTS;
 }
'
```

The next portion contains prior distributions. Stan automatically assigns uninformative priors to all parameters not specified. Since treatment effect sizes in education field trials are rarely greater than 0.5, we specify priors that concentrate most of their weight on values smaller than 0.5. This will help the model avoid more ludicrous estimates that we would not believe anyway.

```
priors <- '
 skipperATE~normal(0,.5);
 firsterATE~normal(0,.5);
'
```

Our Stan code ends with the log-likelihood from (6), (7), and (8).

The models for treatment students' data, usage model (6) and outcome model (7) are fairly standard. They are coded by assigning distributions to observed data `first` and `Ytrt`; Stan then translates these expressions into terms in the log-likelihood. First the usage model:

```
trtMods <- '
 first~bernoulli(piT);
 Ytrt~normal(alphaT+Xtrt*betaY,sigT);
'
```

The outcome model for control students, (8), is a normal mixture model. Writing a mixture model in Stan is somewhat complicated—instead of simply assigning a distribution, we construct this part of the log the log likelihood more directly. In Stan, the variable `target` is the log-likelihood, and we can add terms (8) directly to it. The expression is further complicated because (8) includes a sum of products inside a logarithm. For computational reasons, it is advantageous to take the logarithm of each of the terms of the sum, then exponentiate them, and finally take the logarithm of the sum as a whole. This is accomplished with the `log_sum_exp()` function. A more complete explanation is in the (surprisingly readable) Stan manual, Team (2017), Section 13. This all comes out to:

```
outModCtl <- '
 for(i in 1:nc)
  target += log_sum_exp(
   log(piC[i]) + normal_lpdf(Yctl[i] | alphaCF+Xctl[i,]*betaY,sigCF),
   log((1-piC[i])) + normal_lpdf(Yctl[i] |alphaCS+Xctl[i,]*betaY,sigCS));
} //closes "model{"
'
```

where the term `normal_lpdf(Yctl[i] | alphaCS+Xctl[i,]*betaY,sigCS)`, e.g., represents the logarithm of the normal density function, with mean `alphaCS+Xctl[i,]*betaY` and standard deviation `sigCS`, evaluated at the point `Yctl[i]`.

A useful model checking device, called "posterior predictive check" (Rubin et al., 1984; Gelman et al., 1996), compares real data to random fake data simulated from the fitted posterior distribution. Stan can simulate fake data in an additional model block called `generated quantities`, which we will use to simulate fake values of M_T for control subjects, and fake outcomes for both control and treated subjects. Stan uses functions ending in the suffix `_rng` (i.e. random number generator) to simulate random values from a given distribution:

```
generatedQuantities <- '
generated quantities{
 int<lower=0,upper=1> first_repC[nc]; //M_T for controls
 real Ytrt_rep[nt]; // outcomes for treateds
 real Yctl_rep[nc]; // outcomes for controls

 for(i in 1:nt){
  if(first[i]==1)
   Ytrt_rep[i]=normal_rng(alphaTF+Xtrt[i,]*betaY,sigTF);
  else
   Ytrt_rep[i]=normal_rng(alphaTS+Xtrt[i,]*betaY,sigTS);
 }
 for(i in 1:nc){
  first_repC[i]=bernoulli_rng(piC[i]);
  if(first_repC[i]==1)
   Yctl_rep[i]=normal_rng(alphaCF+Xctl[i,]*betaY,sigCF);
  else
   Yctl_rep[i]=normal_rng(alphaCS+Xctl[i,]*betaY,sigCS);
 }
}
'
```

Finally, we will combine each of these chunks into the file `psMod.stan`:

```
cat(dataBlock,paramBlock,tranParamBlock,modelBlock1,priors,
    trtMods,outModCtl,generatedQuantities,
    file='psMod.stan',sep='\n')
```

6.2. Preparing the Data in R

We will call Stan from within R to fit the model. This is a two-step process: first, re-format the data so that it may be loaded into Stan, then call the `stan()` function, with the data, the location of the Stan code, and fitting parameters as arguments.

We will pass the data to Stan in the form of a list:

```
stanDat <- list()
```

The named elements of the list correspond to the items in the `data` portion of the Stan code. First, the sample sizes:

```
stanDat$nc <- sum(dat$treatment==0)
stanDat$nt <- sum(dat$treatment==1)
```

Next, the outcome data:

```
stanDat$Yctl <- dat$Y[dat$treatment==0]
stanDat$Ytrt <- dat$Y[dat$treatment==1]
```

To construct the covariate matrix, we use the `model.matrix()` function which takes a model formula and a data frame, and returns a numeric matrix, including continuous covariates and dummy variables constructed from factors. We will also remove the intercept column and use the `scale()` function to transform each of the matrix columns into Z-scores by subtracting their means and dividing by their standard deviations. The full covariate matrix is passed to Stan as two matrices, one for treatment students and one for control, along with the number of covariates.

```
X <- model.matrix(~grade+race+sex+frl+spec+esl+pretest, data=dat)
X <- scale(X[,-1])

stanDat$ncov <- ncol(X)
stanDat$Xctl <- X[dat$treatment==0,]
stanDat$Xtrt <- X[dat$treatment==1,]
```

The last element of `stanDat` is the usage data:

```
first <- ifelse(dat$begin=='firster',1,0)
## keep only treatment cases
stanDat$first <- first[dat$treatment==1]
```

6.3. Calling Stan from R

To fit the model, we use the `rstan` package, which can be installed into `R` after Stan has itself been installed on the computer. `install.packages()`, which can be used to install all of the `R` packages we will use in the tutorial, only needs to be run the first time a package is used.

```
install.packages('rstan')
```

`library()`, which must be run in each new `R` session, loads the package:

```
library(rstan)
```

If the computer has sufficient memory, Stan can be run in parallel:

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

Then, the model is fit using the `stan()` command:

```
mod <- stan('psMod.stan',data=stanDat)
```

Additional parameters may be passed to the `stan()` function. The algorithm Stan uses to fit models is a form of Markov Chain Monte Carlo (MCMC) called "Hamiltonian Monte Carlo" (Hoffman and Gelman, 2014). Rather than calculate the posterior distribution analytically, MCMC draws random values from the posterior distribution and uses them to estimate the posterior, or its means, quantiles, and other attributes. MCMC is a Markov chain that draws new values for the parameters, conditional on previous draws—if the model is well-specified and identified, the Markov chain converges to a stationary distribution, which is the true posterior. Therefore, the MCMC process has two parts: first, a "warmup" stage, in which the Markov chain converges to the posterior distribution, and a "sampling" stage, in which we draw successive samples from the posterior distribution, that we can then use to estimate it. The lengths of these stages are controlled by two arguments to the `stan()` function: `iter` is the total number of iterations, and `warmup` is the number of these dedicated to convergence. Unfortunately, there is no way to know at the outset how long the process will take to converge, so we must guess, and then check the fitted model for convergence. As a default, `stan()` sets `iter` as 2000 and `warmup` as half of `iter`, i.e. 1000, leaving 1000 iterations to sample from the posterior.

The Markov chain has to start somewhere, at an "initial point." The `stan()` default is to set these initial points at random, but users may specify them as well. By time the Markov chain converges, the initial point should be irrelevant. This provides an important set of convergence checks: instead of running one Markov chain to estimate the posterior distribution, run several separate Markov chains, each starting at a different initial point, and compare their results—if they disagree, the Markov chain has probably not converged. The number of chains in a Stan model is specified with the `chains` argument, which defaults to four.

We may call the same model as above with the code

```
mod <- stan('psMod.stan',data=stanDat,iter=2000,warmup=1000,chains=4)
```

6.4. Viewing and Interpreting Model Results

To see a summary of model results, use the `print()` or the `summary()` functions:

```
print(mod)
```

This returns a table of results. Depending on the size of the model, this can be rather large and overwhelming—instead, we will print a subset of the results:

```
print(mod,
      pars=c('alphaTF','alphaTS','alphaCF','alphaCS',
           'firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))
```

```
## Inference for Stan model: psMod.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd  2.5% 97.5% n_eff Rhat
## alphaTF     0.03    0.00 0.04 -0.05  0.10  4000    1
## alphaTS     0.04    0.00 0.06 -0.07  0.16  4000    1
## alphaCF    -0.25    0.01 0.10 -0.41 -0.05   383    1
## alphaCS    -0.44    0.01 0.21 -0.82 -0.10   374    1
## firsterATE  0.28    0.00 0.10  0.07  0.45   464    1
## skipperATE  0.48    0.01 0.20  0.15  0.85   419    1
## ATEdiff    -0.21    0.01 0.29 -0.75  0.26   398    1
##
## Samples were drawn using NUTS(diag_e) at Thu Jan 18 16:25:28 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Each row corresponds to a parameter, from either the `parameters` or `transformed parameters` section of the model. There are two types of columns. Columns labeled `se_mean`, `n_eff`, and `Rhat` give diagnostics of the MCMC process; we will discuss these below in Section 7.. The remaining columns summarize each parameter's marginal posterior distribution, giving estimates of the model parameters with uncertainty. The `mean` column is the mean of the marginal posterior, which may be taken as a point estimate. `sd` is the marginal posterior standard deviations, analogous to the standard error in classical models. The `2.5%` and `97.5%` columns estimate 95% credible intervals: the model fit estimates an 95% probability that the parameter is in this interval. Percentiles other than 2.5 and 97.5 may be computed as well or instead, using the `probs` field in `print()`.

Are there effects for firsters and skippers? The posterior distribution of the model parameter `firsterATE` captures our estimate of the ATE for firsters, with uncertainty. Based on the printed output, the posterior for the firster ATE has a mean of 0.28, a standard deviation of 0.1, and a 95% credible interval of [0.07, 0.45]. According to these results, there is a substantial effect of being assigned to CTAI for subjects who would, if assigned, begin on the first section. For skippers, the model estimates an ATE of 0.48, with a posterior standard deviation of 0.2, and a 95% credible interval of [0.15, 0.85]. This is an even bigger effect of assignment to CTAI.

How confident may we be that the effect of assignment to CTAI is larger for skippers than for firsters? The `ATEdiff` parameter estimates the difference of the two average effects. The mean of its posterior—a point estimate of the difference—is simply the difference of the two estimated effects: $0.28-0.48=$-0.21. The posterior standard deviation of the difference between the ATEs is 0.29, and a 95% credible interval is [-0.75, 0.26]. Evidently, the data are consistent with either ATE being larger—though the model estimates a larger effect for skippers, the data are also consistent with a larger effect for firsters.

Each of the estimated ATEs, for skippers and for firsters, is a causal effect—the difference between potential outcomes—and our estimates are unconfounded due to randomization of treatment

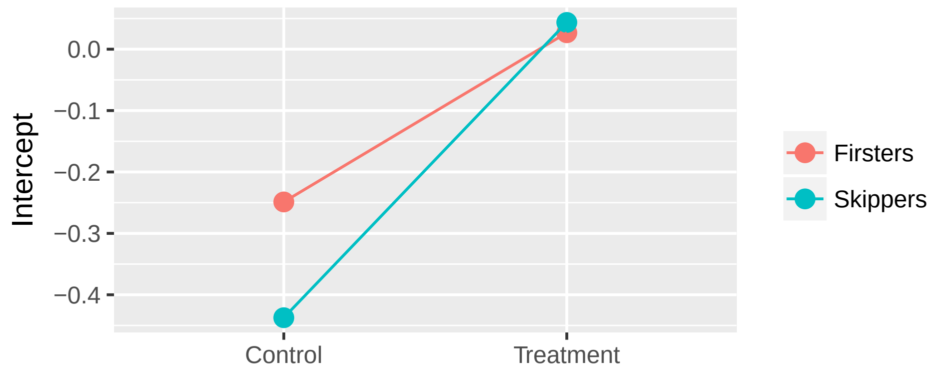


Figure 1: Model results, in terms of estimated intercepts `alphaTF`, `alphaTS`, `alphaCF`, and `alphaCS`.

assignment. The difference between the ATEs, however, is not causal. Nowhere do we model potential outcomes for possible values of M_T (i.e. there is no Y_S or Y_F for the outcomes we would see were subjects skippers or firsters). Also, M_T is not randomly assigned; presumably skippers and firsters varied in more than just the first section they worked, so it is hard to attribute the difference in ATEs to M_T. Nevertheless, the principal effects and their difference can provide insight into CTAI's operation and mechanisms.

The intercept parameters from the model, `alphaTF`, `alphaTS`, `alphaCF`, and `alphaCS`, shed more light on the estimated treatment effects. The two intercepts for the treatment group, `alphaTF` and `alphaTS`, are estimated as 0.03 and 0.04, respectively. They are roughly equal, indicating that there is little difference in performance between firsters and skippers in the control group. On the other hand, `alphaCF` and `alphaCS`, are estimated as -0.25 and -0.44, implying that skippers assigned to control scored substantially lower, on the posttest, than control firsters. While assignment to CTAI evidently boosted posttest scores for both groups, its effect on skippers was larger, allowing those otherwise weaker students to catch up with their peers. This is illustrated schematically in Figure 1. This result also helps emphasize the our reliance on modeling: while `alphaTF` and `alphaTS` are fairly straightforward to estimate, `alphaCF` and `alphaCS`, whose difference comprises almost the entire difference between the estimated ATEs, require the full complex model to estimate. This fact is further reflected in other model results: `alphaCF` and `alphaCS` have higher posterior standard deviations than their treatment group counterparts, implying more uncertainty.

Working directly with the MCMC draws from the posterior gives us even more flexibility. The `extract()` function extracts these draws from the fitted Stan model:

```
draws <- extract(mod)
```

This produces a list whose elements are vectors or matrices MCMC draws for each of the model parameters. The fit summaries returned by the `print()` function above can be replicated from the MCMC draws, for instance,

```r
mean(draws$ATEdiff)

## [1] -0.2054
```

They may also be used to estimate new quantities; for instance, the posterior probability that `firsterATE` is greater than zero may be estimated as:

```r
mean(draws$firsterATE>0)

## [1] 0.9982
```

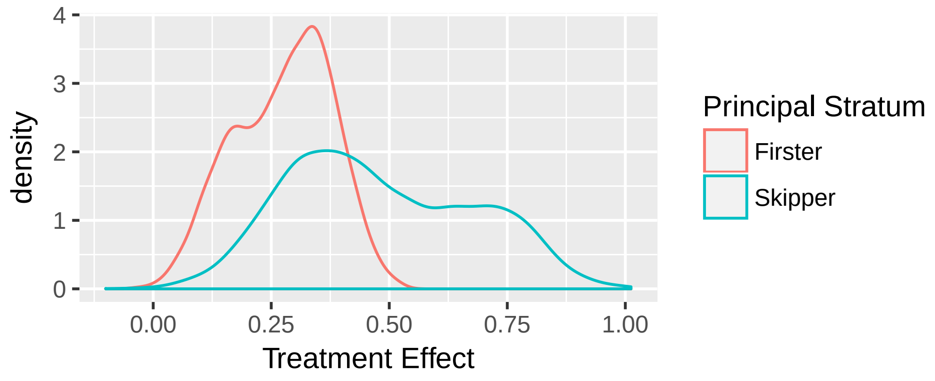
and the probability that the ATE for skippers is greater than the ATE for firsters is estimated as

```r
mean(draws$skipperATE>draws$firsterATE)

## [1] 0.7153
```

They may also be used to plot results, for instance, an estimate of the posterior distributions:

```r
library(ggplot2)
ggplot(mapping=aes(c(draws$firsterATE,draws$skipperATE),
           color=rep(c('Firster','Skipper'),
               each=length(draws$firsterATE))))+
 geom_density()+
 labs(color='Principal Stratum',x='Treatment Effect')
```



The posterior distribution for `skipperATE` shows some evidence of bi-modality, suggesting that there may be two different ways to fit the model to the data, neither of which is clearly better than the other. This is often taken as a sign that the parameter of interest is not strongly identified in the data.

7. Model Checking

The estimates of principal effects in the previous section depended heavily on modeling assumptions. Misspecified models can yield very misleading results, depending, of course, in how and how severely misspecified they are. Moreover, as documented in Feller et al. (2016) and Griffin et al. (2008), PS models are particularly vulnerable to issues of model misspecification, and even estimates from well-specified models can be biased. For this reason, model checking is especially important in the PS context. That said, how best to check PS models, and whether and to what extent model checking techniques address these models' weaknesses, remains an open question.

This section will outline a number of ways to check a PS model's fit: MCMC convergence diagnostics, standard fit checks for Bayesian models, PS model checks using fake data, and robustness checks for PS.

7.1. Checking MCMC Convergence

As described briefly in Section 6.4., MCMC is a powerful technique that can be used to fit arbitrarily complex Bayesian models. It relies on a Markov chain that (when circumstances are right) converges to the posterior distribution. This section will describe ways to check if the Markov chain converged, and assess whether it continued running long enough after convergence to estimate the posterior distribution with sufficient precision.

The `rstan` output from `summary()` or `print()` contains some MCMC diagnostics. Consider our two most important parameters (excluding posterior distribution quantiles for brevity): `skipperATE` and `firsterATE`:

```
summary(mod,pars=c('firsterATE','skipperATE'),probs=c())[[1]]

##                mean   se_mean     sd n_eff  Rhat
## firsterATE 0.2756 0.004833 0.1041 464.3 1.003
## skipperATE 0.4810 0.009633 0.1972 418.9 1.003
```

The `Rhat` column assesses convergence. Recall that we essentially fit the model four separate times, in four "chains," each starting from a different set of initial values. Since the initial values should be irrelevant after convergence, comparing these chains is a good way of checking convergence. The Rhat statistic (Gelman and Rubin, 1992) compares within-chain variance of the chains to differences between them. At convergence, Rhat should be 1; typically values below 1.1 are considered acceptable. The Rhat statistics here, 1.0033 and 1.0033 easily meet this criterion. The function `stan_rhat()` plots a histogram of Rhat statistics from all of the model's parameters. A quicker way to check that they are all sufficiently close to one is look at their maximum:

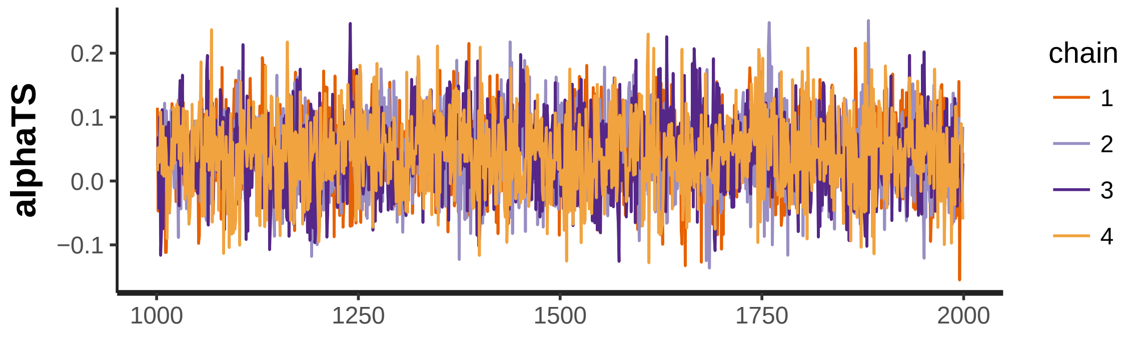
```
max(summary(mod)[[1]][,'Rhat'])

## [1] 1.004
```

This is well below 1.1, indicating convergence.

A similar convergence check is the traceplot. This is a plot of MCMC draws of a parameter, in order and separated by chain. If the pattern of draws differs between chains, or if the mean of the draws in one or more chains appears to be changing over time, the chains may not have converged. Traceplots can be generated in `R` with the `stan_trace()` function:

```r
stan_trace(mod,'alphaTS')
```



In this traceplot, for the parameter `alphaTS`, the four chains are well mixed and no trend is apparent.

The `n_eff` and `se_mean` columns of the `summary()` output measure the adequacy of the of the draws from the posterior distribution. Recall that the true posterior distribution can, theoretically, be calculated exactly from the data and the model—since this is impossible in practice, we instead estimate it via MCMC. The estimates we report are two steps removed from the true ATEs we are trying to estimate: the posterior distribution estimates the true ATEs, and our results estimate the posterior distribution. `n_eff` and `se_mean` measure the accuracy of the second step—our estimates of the posterior. `n_eff` is the "effective sample size" of our sample from the posterior, accounting for the fact that successive MCMC draws are often correlated with each other. Note that this is a distinct (though analogous) quantity from the size of the dataset—it pertains to the quality of our MCMC sample, not to the data themselves. If `n_eff` is too low, we may simply let the model run for more time, and collect a bigger sample from the posterior. `n_eff` for the `firsterATE` parameter is 464.264; this is a large enough sample to estimate at mean, but may or may not be large enough to estimate the bounds of a 95% credible interval (Roy et al., 2016). The `se_mean` column gives the standard error of the estimate of the mean of the posterior distribution—how close is our reported mean to the mean of the true posterior distribution.

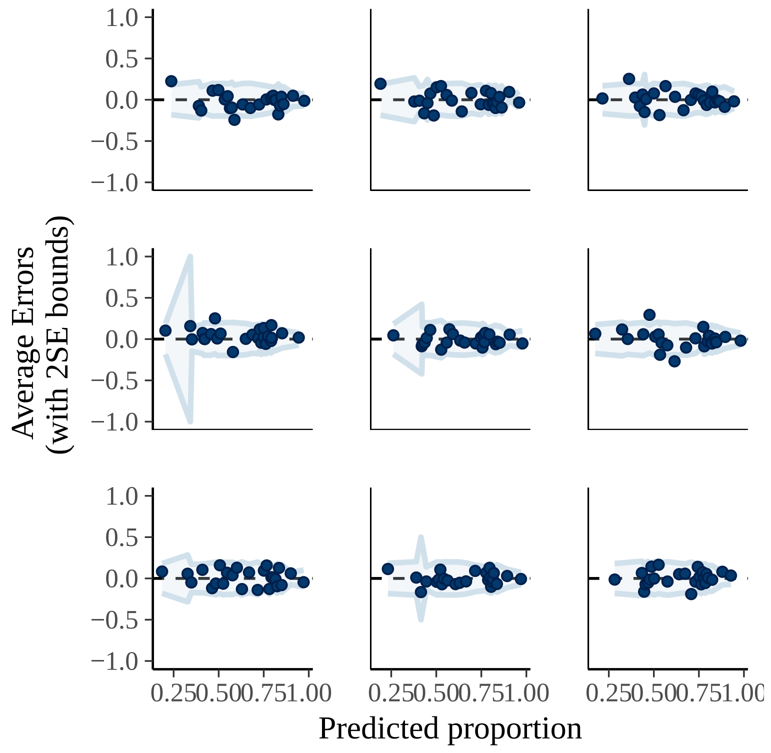
7.2. Standard Model Fit Checks

All of the standard methods to check a Bayesian model's fit to data can be used to check PS models. We will show two here, using the `bayesplot` package in `R` (Gabry, 2016).

```r
library(bayesplot)
```

The first assesses the usage model, by comparing its probability estimates, `piT` (or π_i), to the observed `first` (M_{Ti}). Since `first` is binary, a simple residual plot, comparing `first` to `piT` is very difficult to read and interpret. Instead, a "binned" residual plot Gelman and Hill (2006) bins observations with similar estimated `piT`, and compares their average `piT` to the proportion of ones in their pooled `first`. In the context of a model fit by MCMC, there are many draws of `piT` for each subject. The `ppc_error_binned()` function creates binned residual plots for multiple draws of `piT` (we will do nine).

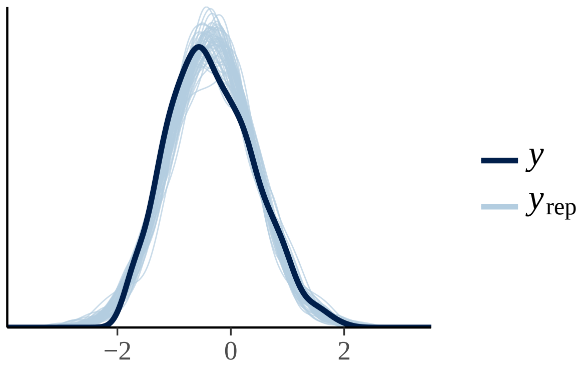
```
ppc_error_binned(stanDat$first,draws$piT[sample(1:4000,9),])
```

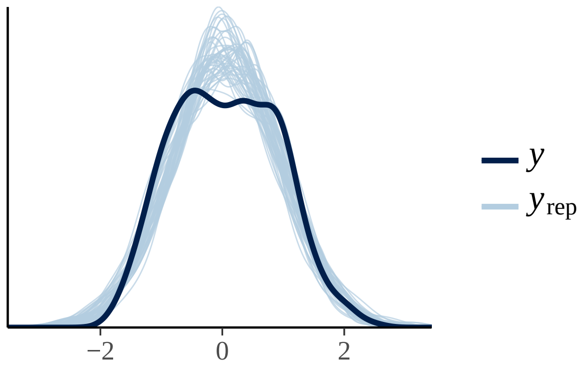


The plot shows no pattern in distribution of the residuals—that is, they appear randomly scattered around zero—and most of the observations fall within the gray lines, as they should in a well-fitting model.

The second model check relies on simulations of outcomes from the PS model fit. Recall, in the `generated quantities` portion of the Stan code in Section 6.1., we instructed Stan to simulate fake outcomes for the the control and treated observations, returning `Yctl_rep` and `Ytrt_rep`. These simulated outcomes can be compared to the true outcomes with the `ppc_dens_overlay()` function from the `bayesplot` package (we are also using the function `grid.arrange()` from the `gridExtra` package to display the plots side-by-side):

```
gridExtra::grid.arrange(
 ppc_dens_overlay(stanDat$Yctl,draws$Yctl_rep[sample(1:4000,50),]),
 ppc_dens_overlay(stanDat$Ytrt,draws$Ytrt_rep[sample(1:4000,50),]),ncol=2)
```





While the model's fit to the control outcomes is reasonably close, the distribution of treatment outcomes flattens in the middle, perhaps due to multi-modality, and the simulated treatment outcomes do not share this trait.

7.3. Running the Model on Fake Data

General fit assessments are extremely useful for constructing a well-fitting model and diagnosing problems. However, no model can fit a dataset perfectly, and it is unclear which fit deficiencies are innocuous and which are problematic. At the same time, even a well-specified model can sometimes yield the wrong result. One way to better understand the model's operating characteristics is to run the model on a fake dataset in which the true parameter values are known. If the model returns the right answer, perhaps despite some model misspecification, our confidence in its results increases.

How should we construct a fake dataset? Ideally, it would mimic salient features of the real data—for instance, the relationships between the covariates, M_T, and Y, and the distribution of the data. Sales and Pane (2017) suggests using real data from the treatment group to construct a fake dataset. Specifically, discard data from control subjects and duplicate the treatment data; label the duplicated treatment data as control.

```r
stanDatF <- within(stanDat,{
    Yctl <- Ytrt
    Xctl <- Xtrt
    nc <- nt})
modNoEff <- stan('psMod.stan',data=stanDatF)
```

The new fake controls are identical to their treatment counterparts; moreover, their "true" values of M_T are known, if unavailable to the model. The model misspecification present in the real treatment group is also present in the fake data. On the other hand, this technique does not assess model misspecification in the control group, and the fact that the treatment and control groups are identical may make model-fitting spuriously easier.

In this fake dataset, there is no treatment effect for firsters or for skippers. Did the model estimate zero treatment effect?

```
summary(modNoEff,c('firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))[[1]]

##                mean  se_mean    sd  2.5%   98% n_eff Rhat
## firsterATE   0.015   0.0037 0.095 -0.15 0.24   640    1
## skipperATE  -0.026   0.0077 0.169 -0.42 0.25   489    1
## ATEdiff      0.042   0.0114 0.248 -0.36 0.65   469    1
```

In this case, the model returned (roughly) the correct answer.

It is also worth checking model results when there is an effect. Here we simulate a random treatment effect for the treatment observations that does not depend on M_T.

```
stanDatF2 <- within(stanDatF,
                    Ytrt <- Ytrt+rnorm(nt,0.35,0.1))
modConstEff <- stan('psMod.stan',data=stanDatF2)
```

The results are encouraging:

```
summary(modConstEff,c('firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))[[1]]

##               mean  se_mean    sd  2.5%   98% n_eff Rhat
## firsterATE 0.361   0.0046 0.094  0.20 0.60   427    1
## skipperATE 0.314   0.0087 0.172 -0.13 0.59   393    1
## ATEdiff    0.047   0.0132 0.250 -0.35 0.71   360    1
```

Here the model returned (roughly) the correct ATE for both groups.

Another scenario to consider is a treatment effect pattern like what our model estimated in Section 6.4.: a larger effect among skippers than among firsters. Instead of adding the treatment effect to the treated subjects, we'll subtract it from the control subjects:

```
stanDatF3 <- within(stanDatF,
                    Yctl <- Yctl-(rnorm(nt,0.25,0.1)+0.2*(1-first)))
modDiffEff <- stan('psMod.stan',data=stanDatF3)
```

The model continues to perform well in this case:

```
summary(modDiffEff,c('firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))[[1]]

##               mean  se_mean   sd  2.5%   98% n_eff Rhat
## firsterATE   0.28   0.0014 0.08  0.12 0.44  3290    1
## skipperATE   0.38   0.0024 0.14  0.10 0.64  3255    1
## ATEdiff     -0.10   0.0036 0.20 -0.48 0.30  3054    1
```

The fact that the model returned the right answer for these three cases is encouraging, though, of course, it does not guarantee that it will return the right answer in all cases.

7.4. ROBUSTNESS CHECKS

Another route to assessing the impact of modeling assumptions on treatment effect estimates is to alter those assumptions and re-estimate the effects. In other words, formulate a new statistical model and check that it returns approximately the same answer. If it does, that is evidence that the tested modeling assumptions are innocuous, or at least no worse than the new alternative.

Each element of the statistical model can and should be checked. Here we will present two examples. One of the most important yet dubious assumptions in the model is the assumption of normally distributed residuals. Feller et al. (2016) replaced a student-t distribution for the normal distribution, estimating the degrees of freedom parameter from the data; we will do the same here.

To re-write the model, we declare two new parameters in the `parameters` model block, `nuT` and `nuC`, for degrees of freedom in the treatment and control conditions, respectively, bounding each parameter below by 1. We also re-write and replace the relevant part of the `model` block, including the gamma prior for degrees of freedom suggested by Juárez and Steel (2010), and save to a new file, `psModRobust.stan`.

```
robustParams <- paste(substr(paramBlock,1,nchar(paramBlock)-2),
 ' real<lower=1> nuT;',' real<lower=1> nuC;','}',sep='\n')

robustModels <- '
nuT ~ gamma(2,0.1); nuC ~ gamma(2,0.1);

first~bernoulli(piT);
Ytrt~student_t(nuT,alphaT+Xtrt*betaY,sigT);

for(i in 1:nc)
 target += log_sum_exp(
  log(piC[i])+
   student_t_lpdf(Yctl[i] | nuC,alphaCF+Xctl[i,]*betaY,sigCF),
  log((1-piC[i]))+
   student_t_lpdf(Yctl[i] | nuC,alphaCS+Xctl[i,]*betaY,sigCS));
'
cat(dataBlock,robustParams,tranParamBlock,modelBlock1,priors,
    robustModels,'}',
    file='psModRobust.stan',sep='\n')
```

```
robMod <- stan('psModRobust.stan',data=stanDat)
```

```
summary(robMod,c('firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))[[1]]

##                 mean  se_mean     sd     2.5%  97.5% n_eff  Rhat
## firsterATE   0.2275 0.003503 0.0995  0.05328 0.4209 806.9 1.001
## skipperATE   0.5780 0.007010 0.1885  0.20008 0.8891 723.3 1.002
## ATEdiff     -0.3505 0.010436 0.2740 -0.79150 0.1870 689.1 1.002
```

The fitted model gave estimates roughly similar to the normal model.

Another important assumption is the linear additive covariate model for Y and for M_T. To relax this assumption, we add all two-way interactions between sex, free and reduced price lunch status, special education status, and pretest and a quadratic pretest term to the model, and re-fit.

```
X2 <- model.matrix(~grade+race+esl+(sex+frl+spec+pretest)^2+I(pretest^2), data=dat)
X2 <- scale(X2[,-1])

stanDat2 <- stanDat
stanDat2$ncov <- ncol(X2)
stanDat2$Xctl <- X2[dat$treatment==0,]
stanDat2$Xtrt <- X2[dat$treatment==1,]
intMod <- stan('psMod.stan',data=stanDat2)
```

```
summary(intMod,c('firsterATE','skipperATE','ATEdiff'),
      probs=c(0.025,0.975))[[1]]

##              mean se_mean   sd   2.5%  98% n_eff Rhat
## firsterATE  0.28  0.0058 0.11  0.058 0.45   337    1
## skipperATE  0.44  0.0115 0.21  0.111 0.85   324    1
## ATEdiff    -0.16  0.0173 0.30 -0.761 0.30   304    1
```

For more examples of robustness checks, see Sales and Pane (2017) and Feller et al. (2016).

Another model checking device not discussed here is the posterior predictive p-value, discussed in Barnard et al. (2003) and elsewhere.

8. Conclusion

Randomized field trials in educational technology research, provide invaluable evidence regarding the effectiveness of educational tools and products. When researchers also collect log data, these studies can also be a source of evidence regarding how educational technology is used in authentic settings. Principal stratification is a method to link those two types of evidence: how does real-life variation in usage relate to variation in treatment effects? Answering those questions can shed light on what it is about educational technology that produces its effects.

PS is a powerful tool, but it can also be difficult to use in practice. This paper is an attempt to guide researchers through the process of estimating principal effects. Our hope is that it will help educational technology researchers delve deeper into their data, and gain new insight into the learning process.

We also hope that as researchers adopt PS, they will continue to develop it. In particular, model checking and validation continue to be areas of open research, especially given the results in Feller et al. (2016). Additionally, PS can be expanded to cover a wider range of scenarios than the simple one described here. For instance, Jin and Rubin (2008) estimates principal effects as a function of a continuous intermediate variable, Mattei et al. (2013) estimates principal effects on multiple outcomes simultaneously, and Sales and Pane (2017) estimates principal effects as a function of a latent intermediate variable. Given the dimension, complexity, and longitudinal structure of log data, PS models incorporating more complex intermediate variables may be particularly useful.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant Number 1420374. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

ANGRIST, J. D., IMBENS, G. W., AND RUBIN, D. B. 1996. Identification of causal effects using instrumental variables. *Journal of the American statistical Association 91,* 434, 444–455.

BARNARD, J., FRANGAKIS, C. E., HILL, J. L., AND RUBIN, D. B. 2003. Principal stratification approach to broken randomized experiments: A case study of school choice vouchers in new york city. *Journal of the American Statistical Association 98,* 462, 299–323.

EFRON, B. ET AL. 1979. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics 7,* 1, 1–26.

ESCUETA, M., QUAN, V., NICKOW, A. J., AND OREOPOULOS, P. 2017. Education technology: an evidence-based review. Tech. rep., National Bureau of Economic Research.

FELLER, A., GREIF, E., MIRATRIX, L., AND PILLAI, N. 2016. Principal stratification in the twilight zone: Weakly separated components in finite mixture models. *arXiv preprint arXiv:1602.06595*.

FELLER, A., GRINDAL, T., MIRATRIX, L., PAGE, L. C., ET AL. 2016. Compared to what? variation in the impacts of early childhood education by alternative care type. *The Annals of Applied Statistics 10,* 3, 1245–1285.

FRANGAKIS, C. E. AND RUBIN, D. B. 2002. Principal stratification in causal inference. *Biometrics 58,* 1, 21–29.

GABRY, J. 2016. *bayesplot: Plotting for Bayesian Models*. R package version 1.1.0.

GELMAN, A., CARLIN, J. B., STERN, H. S., DUNSON, D. B., VEHTARI, A., AND RUBIN, D. B. 2014. *Bayesian data analysis*. Vol. 2. CRC press Boca Raton, FL.

GELMAN, A. AND HILL, J. 2006. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press.

GELMAN, A., JAKULIN, A., PITTAU, M. G., AND SU, Y.-S. 2008. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 1360–1383.

GELMAN, A., LEE, D., AND GUO, J. 2015. Stan: A probabilistic programming language for bayesian inference and optimization. *Journal of Educational and Behavioral Statistics 40,* 5, 530–543.

GELMAN, A., MENG, X.-L., AND STERN, H. 1996. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, 733–760.

GELMAN, A. AND RUBIN, D. B. 1992. Inference from iterative simulation using multiple sequences. *Statistical science*, 457–472.

GELMAN, A. AND SHALIZI, C. R. 2013. Philosophy and the practice of bayesian statistics. *British Journal of Mathematical and Statistical Psychology 66,* 1, 8–38.

GRIFFIN, B. A., MCCAFFERY, D. F., AND MORRAL, A. R. 2008. An application of principal stratification to control for institutionalization at follow-up in studies of substance abuse treatment programs. *The annals of applied statistics 2,* 3, 1034.

HOFFMAN, M. D. AND GELMAN, A. 2014. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research 15,* 1, 1593–1623.

HOLLAND, P. W. 1986. Statistics and causal inference. *Journal of the American statistical Association 81,* 396, 945–960.

JIN, H. AND RUBIN, D. B. 2008. Principal stratification for causal inference with extended partial compliance. *Journal of the American Statistical Association 103,* 481, 101–111.

JUÁREZ, M. A. AND STEEL, M. F. 2010. Model-based clustering of non-gaussian panel data based on skew-t distributions. *Journal of Business & Economic Statistics 28,* 1, 52–66.

KADANE, J. B. 2011. *Principles of uncertainty*. CRC Press.

KRUSCHKE, J. 2014. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

LACHIN, J. M. 2000. Statistical considerations in the intent-to-treat principle. *Controlled clinical trials 21,* 3, 167–189.

LEE, D. S. 2009. Training, wages, and sample selection: Estimating sharp bounds on treatment effects. *The Review of Economic Studies 76,* 3, 1071–1102.

MATTEI, A., LI, F., MEALLI, F., ET AL. 2013. Exploiting multiple outcomes in bayesian principal stratification analysis with application to the evaluation of a job training program. *The Annals of Applied Statistics 7,* 4, 2336–2360.

MIRATRIX, L., FUREY, J., FELLER, A., GRINDAL, T., AND PAGE, L. C. 2017. Bounding, an accessible method for estimating principal causal effects, examined and explained. *arXiv preprint arXiv:1701.03139.*

PAGE, L. C. 2012. Principal stratification as a framework for investigating mediational processes in experimental settings. *Journal of Research on Educational Effectiveness 5,* 3, 215–244.

PANE, J. F., GRIFFIN, B. A., MCCAFFREY, D. F., AND KARAM, R. 2014. Effectiveness of cognitive tutor algebra i at scale. *Educational Evaluation and Policy Analysis 36,* 2, 127–144.

PARADIS, E. 2002. R for beginners.

R CORE TEAM. 2016. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

ROY, P., LAPRISE, R., AND GACHON, P. 2016. Sampling errors of quantile estimations from finite samples of data. *arXiv preprint arXiv:1610.03458.*

RUBIN, D. B. 1978. Bayesian inference for causal effects: The role of randomization. *The Annals of statistics*, 34–58.

RUBIN, D. B. ET AL. 1984. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics 12,* 4, 1151–1172.

SALES, A. C. AND PANE, J. F. 2017. The role of mastery learning in intelligent tutoring systems: Principal stratification on a latent variable. *arXiv preprint arXiv:1707.09308*.

STAN DEVELOPMENT TEAM. 2016. RStan: the R interface to Stan. R package version 2.14.1.

TEAM, S. D. 2017. Stan modeling language users guide and reference manual, version 2.17.0.