Statistical Tools for Causal Inference

The SKY Community

2019-09-27

Contents

Introduction

Tools of causal inference are the basic statistical building block behind most scientific results. It is thus extremely useful to have an open source collectively aggreed upon resource presenting and assessing them, as well as listing the current unresolved issues. The content of this book covers the basic theoretical knowledge and technical skills required for implementing staistical methods of causal inference. This means:

- Understanding of the basic language to encode causality,
- Knowledge of the fundamental problems of inference and the biases of intuitive estimators,
- Understanding of how econometric methods recover treatment effects,
- Ability to compute these estimators along with an estimate of their precision using the statistical software R.

This book is geared for teaching causal inference to graduate students that want to apply statistical tools of causal inference. The demonstration of theoretical results are provided, but the final goal is not to have students reproduce them, but mostly to enable them to grasp a better understanding of the fundations for the tools that they will be using. The focus is on understanding the issues and solutions more than understanding the maths that are behind, even though the maths are there and are used to convey the notions rigorously. All the notions and estimators are introduced using a numerical example and simulations, so that each notion is illustrated and appears more intuitive to the students. The second version of this book will contain examples using real applications. The third version will contain exercises.

This book is written in Rmarkdown using the bookdown package. It is available both as a web-book and as a pdf book.

This book is a collaborative effort that is part of the Social Science Knowledge Accumulation Initiative (SKY). The code behind this book is publically available on GitHub and you can propose corrections and updates. How to make contributions to this book is explained on the SKY website. Do not hesitate to make suggestions, modifications and extensions. This way this book will grow and become the living open source collaborative reference for methodological work that it could be.

Part I

Fundamental Problems of Inference

Introduction: the Two Fundamental Problems of Inference

When trying to estimate the effect of a program on an outcome, we face two very important and difficult problems: the Fundamental Problem of Causal Inference (FPCI) and the Fundamental Problem of Statistical Inference (FPSI).

In its most basic form, the FPCI states that our causal parameter of interest (TT, short for Treatment on the Treated, that we will define shortly) is fundamentally unobservable, even when the sample size is infinite. The main reason for that is that one component of TT, the outcome of the treated had they not received the program, remains unobservable. We call this outcome a counterfactual outcome. The FPCI is a very dispiriting result, and is actually the basis for all of the statistical methods of causal inference. All of these methods try to find ways to estimate the counterfactual by using observable quantities that hopefully approximate it as well as possible. Most people, including us but also policymakers, generally rely on intuitive quantities in order to generate the counterfactual (the individuals without the program or the individuals before the program was implemented). Unfortunately, these approximations are generally very crude, and the resulting estimators of TT are generally biased, sometimes severely.

The Fundamental Problem of Statistical Inference (FPSI) states that, even if we have an estimator E that identifies TT in the population, we cannot observe E because we only have access to a finite sample of the population. The only thing that we can form from the sample is a sample equivalent $\hat{E}$ to the population quantity E, and $\hat{E} \neq E$. Why is $\hat{E} \neq E$? Because a finite sample is never perfectly representative of the population. What can we do to deal with the FPSI? I am going to argue that there are mainly two things that we might want to do: estimating the extent of sampling noise and decreasing sampling noise.

Chapter 1

Fundamental Problem of Causal Inference

In order to state the FPCI, we are going to describe the basic language to encode causality set up by Rubin, and named Rubin Causal Model (RCM). RCM being about partly observed random variables, it is hard to make these notions concrete with real data. That's why we are going to use simulations from a simple model in order to make it clear how these variables are generated. The second virtue of this model is that it is going to make it clear the source of selection into the treatment. This is going to be useful when understanding biases of intuitive comparisons, but also to discuss the methods of causal inference. A third virtue of this approach is that it makes clear the connexion between the treatment effects literature and models. Finally, a fourth reason that it is useful is that it is going to give us a source of sampling variation that we are going to use to visualize and explore the properties of our estimators.

I use X_i to denote random variable X all along the notes. I assume that we have access to a sample of N observations indexed by $i \in \{1, \ldots, N\}$. "i" will denote the basic sampling units when we are in a sample, and a basic element of the probability space when we are in populations. Introducing rigorous measure-theoretic notations for the population is feasible but is not necessary for comprehension.

When the sample size is infinite, we say that we have a population. A population is a very useful fiction for two reasons. First, in a population, there is no sampling noise: we observe an infinite amount of observations, and our estimators are infinitely precise. This is useful to study phenomena independently of sampling noise. For example, it is in general easier to prove that an estimator is equal to TT under some conditions in the population. Second, we are most of the time much more interested in estimating the values of parameters in the population rather than in the sample. The population parameter, independent of sampling noise, gives a much better idea of the causal parameter for the population of interest than the parameter in the sample. In general, the estimator for both quantities will be the same, but the estimators for the effetc of sampling noise on these estimators will differ. Sampling noise for the population parameter will generally be larger, since it is affected by another source of variability (sample choice).

1.1 Rubin Causal Model

RCM is made of three distinct building blocks: a treatment allocation rule, that decides who receives the treatment; potential outcomes, that measure how each individual reacts to the treatment; the switching equation that relates potential outcomes to observed outcomes through the allocation rule.

1.1.1 Treatment allocation rule

The first building block of RCM is the treatment allocation rule. Throughout this class, we are going to be interested in inferring the causal effect of only one treatment with respect to a control condition. Extensions to multi-valued treatments are in general self-explanatory.

In RCM, treatment allocation is captured by the variable D_i. $D_i = 1$ if unit i receives the treatment and $D_i = 0$ if unit i does not receive the treatment and thus remains in the control condition.

The treatment allocation rule is critical for several reasons. First, because it switches the treatment on or off for each unit, it is going to be at the source of the FPCI. Second, the specific properties of the treatment allocatoin rule are going to matter for the feasibility and bias of the various econometric methods that we are going to study.

Let's take a few examples of allocation rules. These allocation rules are just examples. They do not cover the space of all possible allocation rules. They are especially useful as concrete devices to understand the sources of biases and the nature of the allocation rule. In reality, there exists even more complex allocation rules (awareness, eligibility, application, acceptance, active participation). Awareness seems especially important for program participation and has only been tackled recently by economists.

First, some notation. Let's imagine a treatment that is given to individuals. Whether each individual receives the treatment partly depends on the level of her outcome before receiving the treatment. Let's denote this variable Y_i^B, with B standing for "Before". It can be the health status assessed by a professional before deciding to give a drug to a patient. It can be the poverty level of a household used to assess its eligibilty to a cash transfer program.

1.1.1.1 Sharp cutoff rule

The sharp cutoff rule means that everyone below some threshold $\bar{Y}$ is going to receive the treatment. Everyone whose outcome before the treatment lies above $\bar{Y}$ does not receive the treatment. Such rules can be found in reality in a lot of situations. They might be generated by administrative rules. One very simple way to model this rule is as follows:

$$D_i = \mathbb{1}[Y_i^B \leq \bar{Y}], \tag{1.1}$$

where $\mathbb{1}[A]$ is the indicator function, taking value 1 when A is true and 0 otherwise.

Example 1.1 (Sharp cutoff rule)**.** Imagine that $Y_i^B = \exp(y_i^B)$, with $y_i^B = \mu_i + U_i^B$, $\mu_i \sim \mathcal{N}(\bar{\mu}, \sigma_\mu^2)$ and $U_i^B \sim \mathcal{N}(0, \sigma_U^2)$. Now, let's choose some values for these parameters so that we can generate a sample of individuals and allocate the treatment among them. I'm going to switch to R for that.

```
param <- c(8,.5,.28,1500)
names(param) <- c("barmu","sigma2mu","sigma2U","barY")
param
```

```
##    barmu sigma2mu  sigma2U     barY
##     8.00     0.50     0.28  1500.00
```

Now, I have choosen values for the parameters in my model. For example, $\bar{\mu} = 8$ and $\bar{Y} = 1500$. What remains to be done is to generate Y_i^B and then D_i. For this, I have to choose a sample size ($N = 1000$) and then generate the shocks from a normal.

```
# for reproducibility, I choose a seed that will give me the same random sample each time I run the pro
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
```

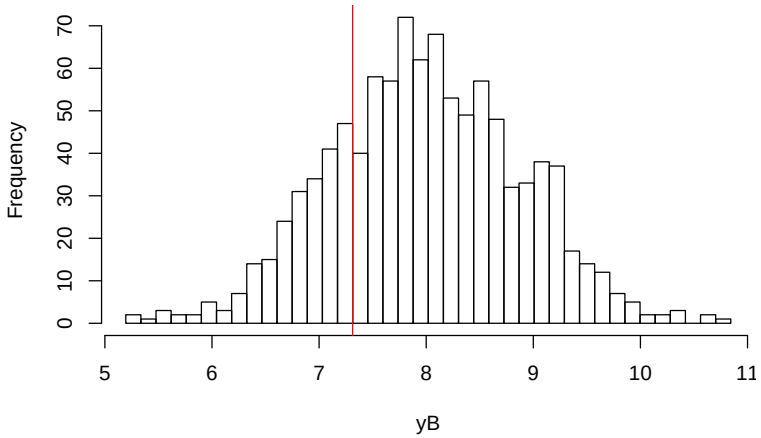


Figure 1.1: Histogram of y_B

Table 1.1: Treatment allocation with sharp cutoff rule

0	769
1	231

```
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- ifelse(YB<=param["barY"],1,0)
```

Let's now build a histogram of the data that we have just generated.

```
# building histogram of yB with cutoff point at ybar
# Number of steps
Nsteps.1 <- 15
#step width
step.1 <- (log(param["barY"])-min(yB[Ds==1]))/Nsteps.1
Nsteps.0 <- (-log(param["barY"])+max(yB[Ds==0]))/step.1
breaks <- cumsum(c(min(yB[Ds==1]),c(rep(step.1,Nsteps.1+Nsteps.0+1))))
hist(yB,breaks=breaks,main="")
abline(v=log(param["barY"]),col="red")
```

You can see on Figure 1.1 a histogram of y_i^B with the red line indicating the cutoff point: $\bar{y} = \ln(\bar{Y}) = 7.3$. All the observations below the red line are treated according to the sharp rule while all the one located above are not. In order to see how many observations eventually receive the treatment with this allocation rule, let's build a contingency table.

```
table.D.sharp <- as.matrix(table(Ds))
knitr::kable(table.D.sharp,caption='Treatment allocation with sharp cutoff rule',booktabs=TRUE)
```

We can see on Table 1.1 that there are 231 treated observations.

1.1.1.2 Fuzzy cutoff rule

This rule is less sharp than the sharp cutoff rule. Here, other criteria than Y_i^B enter into the decision to allocate the treatment. The doctor might measure the health status of a patient following official guidelines, but he might also measure other factors that will also influence his decision of giving the drug to the patient.

The officials administering a program might measure the official income level of a household, but they might also consider other features of the household situation when deciding to enroll the household into the program or not. If these additional criteria are unobserved to the econometrician, then we have the fuzzy cutoff rule. A very simple way to model this rule is as follows:

$$D_i = \mathbb{1}[Y_i^B + V_i \leq \bar{Y}], \tag{1.2}$$

where V_i is a random variable unobserved to the econometrician and standing for the other influences that might drive the allocation of the treatment. V_i is distributed according to a, for the moment, unspecified cumulative distribution function F_V. When V_i is degenerate (*i.e.* it has only one point of support: it is a constant), the fuzzy cutoff rule becomes the sharp cutoff rule.

1.1.1.3 Eligibility + self-selection rule

It is also possible that households, once they have been made eligible to the treatment, can decide whether they want to receive it or not. A patient might be able to refuse the drug that the doctor suggests she should take. A household might refuse to participate in a cash transfer program to which it has been made eligible. Not all programs have this feature, but most of them have some room for decisions by the agents themselves of whether they want to receive the treatment or not. One simple way to model this rule is as follows:

$$D_i = \mathbb{1}[D_i^* \geq 0]E_i, \tag{1.3}$$

where D_i^* is individual i's valuation of the treatment and E_i is whether or not she is deemed eligible for the treatment. E_i might be choosen according to the sharp cutoff rule of to the fuzzy cutoff rule, or to any other eligibility rule. We will be more explicit about D_i^* in what follows.

SIMULATIONS ARE MISSING FOR THESE LAST TWO RULES

1.1.2 Potential outcomes

The second main building block of RCM are potential outcomes. Let's say that we are interested in the effect of a treatment on an outcome Y. Each unit i can thus be in two potential states: treated or non treated. Before the allocation of the treatment is decided, both of these states are feasible for each unit.

Definition 1.1 (Potential outcomes)**.** For each unit i, we define two potential outcomes:

- Y_i^1: the outcome that unit i is going to have if it receives the treatment,
- Y_i^0: the outcome that unit i is going to have if it **does not** receive the treatment.

Example 1.2. Let's choose functional forms for our potential outcomes. For simplicity, all lower case letters will denote log outcomes. $y_i^0 = \mu_i + \delta + U_i^0$, with δ a time shock common to all the observations and $U_i^0 = \rho U_i^B + \epsilon_i$, with $|\rho| < 1$. In the absence of the treatment, part of the shocks U_i^B that the individuals experienced in the previous period persist, while some part vanish. $y_i^1 = y_i^0 + \bar{\alpha} + \theta\mu_i + \eta_i$. In order to generate the potential outcomes, one has to define the laws for the shocks and to choose parameter values. Let's assume that $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ and $\eta_i \sim \mathcal{N}(0, \sigma_\eta^2)$. Now let's choose some parameter values:

```
l <- length(param)
param <- c(param,0.9,0.01,0.05,0.05,0.05,0.1)
names(param)[(l+1):length(param)] <- c("rho","theta","sigma2epsilon","sigma2eta","delta","baralpha")
param
```

```
##         barmu      sigma2mu       sigma2U          barY           rho
##          8.00          0.50          0.28       1500.00          0.90
```

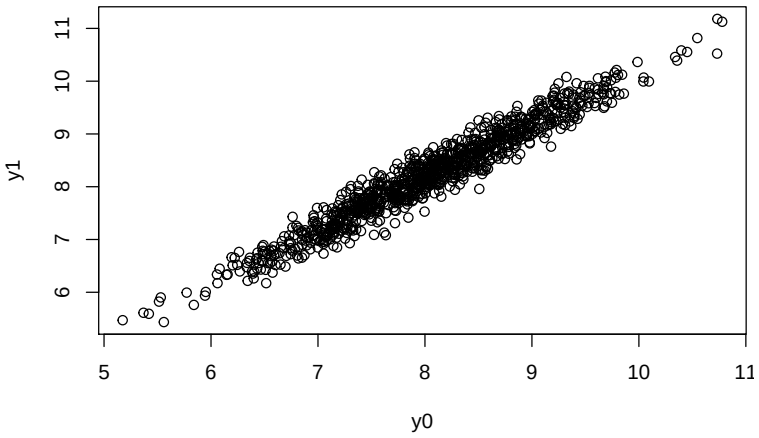


Figure 1.2: Potential outcomes

```
##         theta sigma2epsilon     sigma2eta         delta      baralpha
##          0.01          0.05          0.05          0.05          0.10
```

We can finally generate the potential outcomes;

```
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
```

Now, I would like to visualize my potential outcomes:

```
plot(y0,y1)
```

You can see on the resulting Figure 1.2 that both potential outcomes are positively correlated. Those with a large potential outcome when untreated (*e.g.* in good health without the treatment) also have a positive health with the treatment. It is also true that individuals with bad health in the absence of the treatment also have bad health with the treatment.

1.1.3 Switching equation

The last building block of RCM is the switching equation. It links the observed outcome to the potential outcomes through the allocation rule:

$$\begin{aligned} Y_i &= \begin{cases} Y_i^1 & \text{if } D_i = 1 \\ Y_i^0 & \text{if } D_i = 0 \end{cases} \\ &= Y_i^1 D_i + Y_i^0 (1 - D_i) \end{aligned} \tag{1.4}$$

Example 1.3. In order to generate observed outcomes in our numerical example, we simply have to enforce the switching equation:

```
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

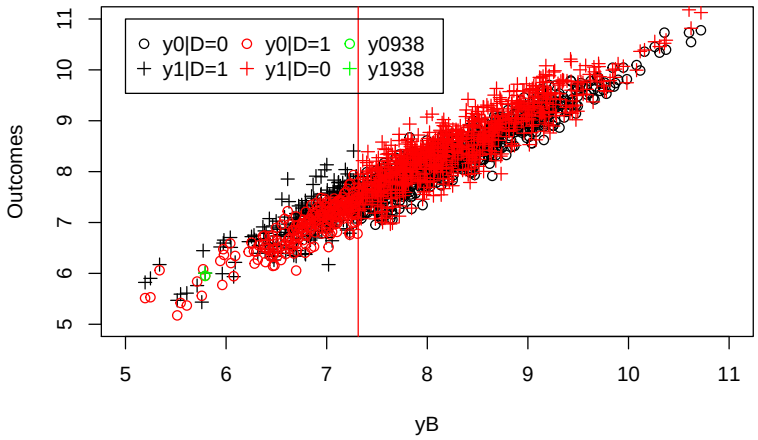


Figure 1.3: Potential outcomes

What the switching equation (1.4) means is that, for each individual i, we get to observe only one of the two potential outcomes. When individual i belongs to the treatment group (*i.e.* $D_i = 1$), we get to observe Y_i^1. When individual i belongs to the control group (*i.e.* $D_i = 0$), we get to observe Y_i^0. Because the same individual cannot be at the same time in both groups, we can NEVER see both potential outcomes for the same individual at the same time.

For each of the individuals, one of the two potential outcomes is unobserved. We say that it is a **counterfactual**. A counterfactual quantity is a quantity that is, according to Hume's definition, contrary to the observed facts. A counterfactual cannot be observed, but it can be conceived by an effort of reason: it is the consequence of what would have happened had some action not been taken.
Remark. One very nice way of visualising the switching equation has been proposed by Jerzy Neyman in a 1923 prescient paper. Neyman proposes to imagine two urns, each one filled with N balls. One urn is the treatment urn and contains balls with the id of the unit and the value of its potential outcome Y_i^1. The other urn is the control urn, and it contains balls with the value of the potential outcome Y_i^0 for each unit i. Following the allocation rule D_i, we decide whether unit i is in the treatment or control group. When unit i is in the treatment group, we take the corresponding ball from the first urn and observe the potential outcome on it. But, at the same time, the urns are connected so that the corresponding ball with the potential outcome of unit i in the control urn disappears as soon as we draw ball i from the treatment urn.

The switching equation works a lot like Schrodinger's cat paradox. Schrodinger's cat is placed in a sealed box and receives a dose of poison when an atom emits a radiation. As long as the box is sealed, there is no way we can know whether the cat is dead or alive. When we open the box, we observe either a dead cat or a living cat, but we cannot observe the cat both alive and dead at the same time. The switching equation is like opening the box, it collapses the observed outcome into one of the two potential ones.
Example 1.4. One way to visualize the inner workings of the switching equation is to plot the potential outcomes along with the criteria driving the allocation rule. In our simple example, it simply amounts to plotting observed (y_i) and potential outcomes (y_i^1 and y_i^0) along y_i^B.

```
plot(yB[Ds==0],y0[Ds==0],pch=1,xlim=c(5,11),ylim=c(5,11),xlab="yB",ylab="Outcomes")
points(yB[Ds==1],y1[Ds==1],pch=3)
points(yB[Ds==0],y1[Ds==0],pch=3,col='red')
points(yB[Ds==1],y0[Ds==1],pch=1,col='red')
test <- 5.8
i.test <- which(abs(yB-test)==min(abs(yB-test)))
points(yB[abs(yB-test)==min(abs(yB-test))],y1[abs(yB-test)==min(abs(yB-test))],col='green',pch=3)
points(yB[abs(yB-test)==min(abs(yB-test))],y0[abs(yB-test)==min(abs(yB-test))],col='green')
abline(v=log(param["barY"]),col="red")
legend(5,11,c('y0|D=0','y1|D=1','y0|D=1','y1|D=0',paste('y0',i.test,sep=''),paste('y1',i.test,sep='')),
```

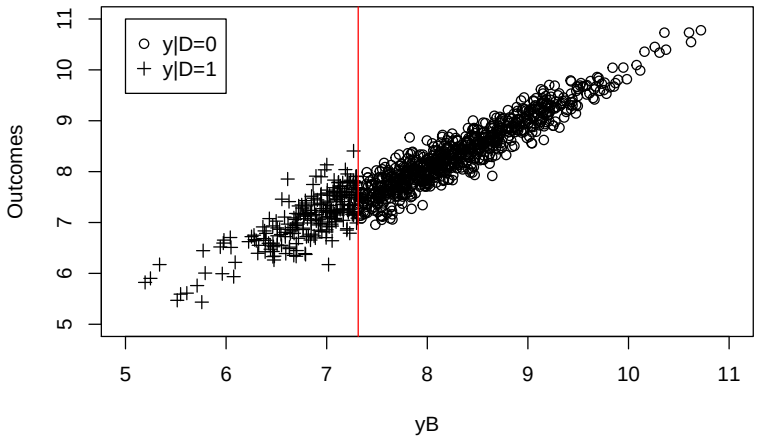


Figure 1.4: Observed outcomes

```
plot(yB[Ds==0],y0[Ds==0],pch=1,xlim=c(5,11),ylim=c(5,11),xlab="yB",ylab="Outcomes")
points(yB[Ds==1],y1[Ds==1],pch=3)
legend(5,11,c('y|D=0','y|D=1'),pch=c(1,3))
abline(v=log(param["barY"]),col="red")
```

Figure 1.3 plots the observed outcomes y_i along with the unobserved potential outcomes. Figure 1.3 shows that each individual in the sample is endowed with two potential outcomes, represented by a circle and a cross. Figure 1.4 plots the observed outcomes y_i that results from applying the switching equation. Only one of the two potential outcomes is observed (the cross for the treated group and the circle for the untreated group) and the other is not. The observed sample in Figure 1.4 only shows observed outcomes, and is thus silent on the values of the missing potential outcomes.

1.2 Treatment effects

RCM enables the definition of causal effects at the individual level. In practice though, we generally focus on a summary measure: the effect of the treatment on the treated.

1.2.1 Individual level treatment effects

Potential outcomes enable us to define the central notion of causal inference: the causal effect, also labelled the treatment effect, which is the difference between the two potential outcomes.

Definition 1.2 (Individual level treatment effect)**.** For each unit i, the causal effect of the treatment on outcome Y is: $\Delta_i^Y = Y_i^1 - Y_i^0$.

Example 1.5. The individual level causal effect in log terms is: $\Delta_i^y = \alpha_i = \bar{\alpha} + \theta\mu_i + \eta_i$. The effect is the sum of a part common to all individuals, a part correlated with μ_i: the treatment might have a larger or a smaller effect depending on the unobserved permanent ability or health status of individuals, and a random shock. It is possible to make the effect of the treatment to depend on U_i^B also, but it would complicate the model.

In Figure 1.3, the individual level treatment effects are the differences between each cross and its corresponding circle. For example, for observation 938, the two potential outcomes appear in green in Figure 1.3. The effect of the treatment on unit 938 is equal to:

$$\Delta_{938}^y = y_{938}^1 - y_{938}^0 = 6.01 - 5.95 = 0.05.$$

Since observation 938 belongs to the treatment group, we can only observe the potential outcome in the presence of the treatment, y^1_{938}.

RCM allows for heterogeneity of treatment effects. The treatment has a large effect on some units and a much smaller effect on other units. We can even have some units that benefit from the treatment and some units that are harmed by the treatment. The individual level effect of the treatment is itself a random variable (and not a fixed parameter). It has a distribution, F_{Δ^Y}.

Heterogeneity of treatment effects seems very natural: the treatment might interact with individuals' different backgrounds. The effect of a drug might depend on the genetic background of an individual. An education program might only work for children that already have sufficient non-cognitive skills, and thus might depend in turn on family background. An environmental regulation or a behavioral intervention might only trigger reactions by already environmentally aware individuals. A CCT might have a larger effect when indiviuals are credit-constrained or face shocks.

Example 1.6. In our numerical example, the distribution of $\Delta^y_i = \alpha_i$ is a normal: $\alpha_i \sim \mathcal{N}(\bar{\alpha}+\theta\bar{\mu}, \theta^2\sigma^2_\mu + \sigma^2_\eta)$. We would like to visualize treatment effect heterogeneity. For that, we can build a histogram of the individual level causal effect.

On top of the histogram, we can also draw the theoretical distribution of the treatment effect: a normal with mean 0.18 and variance 0.05.

```
hist(alpha,main="",prob=TRUE)
curve(dnorm(x, mean=(param["baralpha"]+param["theta"]*param["barmu"]), sd=sqrt(param["theta"]^2*param["
```

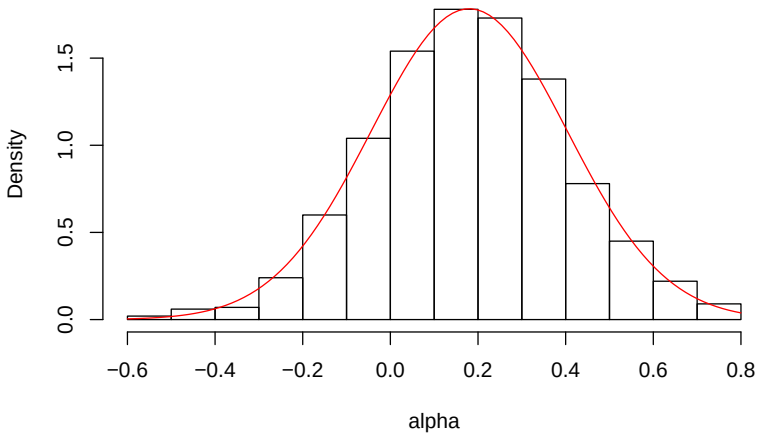


Figure 1.5: Histogram of Δ^y

The first thing that we can see on Figure 1.5 is that the theoretical and the empirical distributions nicely align with each other. We also see that the majority of the observations lies to the right of zero: most people experience a positive effect of the treatment. But there are some individuals that do not benefit from the treatment: the effect of the treatment on them is negative.

1.2.2 Average treatment effect on the treated

We do not generally estimate individual-level treatment effects. We generally look for summary statistics of the effect of the treatment. By far the most widely reported causal parameter is the Treatment on the Treated parameter (TT). It can be defined in the sample at hand or in the population.

Definition 1.3 (Average and expected treatment effects on the treated)**.** The Treatment on the Treated parameters for outcome Y are:

- The average Treatment effect on the Treated in the sample:

$$\Delta^Y_{TT_s} = \frac{1}{\sum_{i=1}^N D_i} \sum_{i=1}^N (Y_i^1 - Y_i^0) D_i,$$

- The expected Treatment effect on the Treated in the population:

$$\Delta^Y_{TT} = \mathbb{E}[Y_i^1 - Y_i^0 | D_i = 1].$$

The TT parameters measure the average effect of the treatment on those who actually take it, either in the sample at hand or in the popluation. It is generally considered to be the most policy-relevant parameter since it measures the effect of the treatment as it has actually been allocated. For example, the expected causal effect on the overall population is only relevant if policymakers are considering implementing the treatment even on those who have not been selected to receive it. For a drug or an anti-poverty program, it would mean giving the treatment to healthy or rich people, which would make little sense.

TT does not say anything about how the effect of the treatment is distributed in the population or in the sample. TT does not account for the heterogneity of treatment effects. In Lecture 7, we will look at other parameters of interest that look more closely into how the effect of the treatment is distributed.
Example 1.7. The value of TT in our sample is:

$$\Delta^y_{TT_s} = 0.158.$$

Computing the population value of TT is slightly more involved: we have to use the formula for the conditional expectation of a censored bivariate normal random variable:

$$\begin{aligned}
\Delta^y_{TT} &= \mathbb{E}[\alpha_i | D_i = 1] \\
&= \bar{\alpha} + \theta \mathbb{E}[\mu_i | \mu_i + U_i^B \leq \bar{y}] \\
&= \bar{\alpha} + \theta \left(\bar{\mu} - \frac{\sigma_\mu^2}{\sqrt{\sigma_\mu^2 + \sigma_U^2}} \frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)} \right) \\
&= \bar{\alpha} + \theta\bar{\mu} - \theta \left(\frac{\sigma_\mu^2}{\sqrt{\sigma_\mu^2 + \sigma_U^2}} \frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)} \right),
\end{aligned}$$

where ϕ and Φ are respectively the density and the cumulative distribution functions of the standard normal. The second equality follows from the definition of α_i and D_i and from the fact that η_i is independent from μ_i and U_i^B. The third equality comes from the formula for the expectation of a censored bivariate normal random variable. In order to compute the population value of TT easily for different sets of parameter values, let's write a function in R:

```
delta.y.tt <- function(param){return(param["baralpha"]+param["theta"]*param["barmu"]
                                     -param["theta"]*((param["sigma2mu"]*dnorm((log(param["barY"])-para
                                                      /(sqrt(param["sigma2mu"]+param["sigma2U"])
                                                        *pnorm((log(param["barY"])-param["barmu"])/(sqr
```

The population value of TT computed using this function is: $\Delta^y_{TT} = 0.172$. We can see that the values of TT in the sample and in the population differ slightly. This is because of sampling noise: the units in the sample are not perfectly representative of the units in the population.

1.3 Fundamental problem of causal inference

At least in this lecture, causal inference is about trying to infer TT, either in the sample or in the population. The FPCI states that it is impossible to directly observe TT because one part of it remains fundamentally unobserved.

Theorem 1.1 (Fundamental problem of causal inference)**.** *It is impossible to observe TT, either in the population or in the sample.*

Proof. The proof of the FPCI is rather straightforward. Let me start with the sample TT:

$$\begin{aligned}\Delta^Y_{TT_s} &= \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N (Y_i^1 - Y_i^0)D_i \\ &= \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^1 D_i - \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^0 D_i \\ &= \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i D_i - \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^0 D_i.\end{aligned}$$

Since Y_i^0 is unobserved whenever $D_i = 1$, $\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^0 D_i$ is unobserved, and so is $\Delta^Y_{TT_s}$. The same is true for the population TT:

$$\begin{aligned}\Delta^Y_{TT} &= \mathbb{E}[Y_i^1 - Y_i^0 | D_i = 1] \\ &= \mathbb{E}[Y_i^1 | D_i = 1] - \mathbb{E}[Y_i^0 | D_i = 1] \\ &= \mathbb{E}[Y_i | D_i = 1] - \mathbb{E}[Y_i^0 | D_i = 1].\end{aligned}$$

$\mathbb{E}[Y_i^0 | D_i = 1]$ is unobserved, and so is Δ^Y_{TT}. □

The key insight in order to understand the FPCI is to see that the outcomes of the treated units had they not been treated are unobservable, and so is their average or expectation. We say that they are counterfactual, contrary to what has happened.

Definition 1.4 (Couterfactual)**.** Both $\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^0 D_i$ and $\mathbb{E}[Y_i^0 | D_i = 1]$ are counterfactual quantities that we will never get to observe.

Example 1.8. The average counterfactual outcome of the treated is the mean of the red circles in the y axis on Figure 1.3:

$$\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N y_i^0 D_i = 6.93.$$

Remember that we can estimate this quantity only because we have generated the data ourselves. In real life, this quantity is hopelessly unobserved.

$\mathbb{E}[y_i^0 | D_i = 1]$ can be computed using the formula for the expectation of a censored normal random variable:

$$\begin{aligned}\mathbb{E}[y_i^0|D_i=1] &= \mathbb{E}[\mu_i+\delta+U_i^0|D_i=1]\\ &= \mathbb{E}[\mu_i+\delta+\rho U_i^B+\epsilon_i|D_i=1]\\ &= \delta+\mathbb{E}[\mu_i+\rho U_i^B|y_i^B\leq\bar{y}]\\ &= \delta+\bar{\mu}-\frac{\sigma_\mu^2+\rho\sigma_U^2}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}.\end{aligned}$$

We can write a function in R to compute this value:

```
esp.y0.D1 <- function(param){
  return(param["delta"]+param["barmu"]
         -((param["sigma2mu"]+param["rho"]*param["sigma2U"])
           *dnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"]))))
         /(sqrt(param["sigma2mu"]+param["sigma2U"])*pnorm((log(param["barY"])-param["barmu"])
                                                          /(sqrt(param["sigma2mu"]+param["sigma2U"])))))
}
```

The population value of TT computed using this function is: $\mathbb{E}[y_i^0|D_i=1]=6.9$.

1.4 Intuitive estimators, confounding factors and selection bias

In this section, we are going to examine the properties of two intuitive comparisons that laypeople, policymakers but also ourselves make in order to estimate causal effects: the with/wihtout comparison (WW) and the before/after comparison (BA). WW compares the average outcomes of the treated individuals with those of the untreated individuals. BA compares the average outcomes of the treated after taking the treatment to their average outcomes before they took the treatment. These comparisons try to proxy for the expected counterfactual outcome in the treated group by using an observed quantity. WW uses the expected outcome of the untreated individuals as a proxy. BA uses the expected outcome of the treated before they take the treatment as a proxy.

Unfortunately, both of these proxies are generally poor and provide biased estimates of TT. The reason that these proxies are poor is that the treatment is not the only factor that differentiates the treated group from the groups used to form the proxy. The intuitive comparisons are biased because factors, other than the treatment, are correlated to its allocation. The factors that bias the intuitive comparisons are generally called confouding factors or confounders.

The treatment effect measures the effect of a ceteris paribus change in treatment status, while the intuitive comparisons capture both the effect of this change and that of other correlated changes that spuriously contaminate the comparison. Intuitive comparisons measure correlations while treatment effects measure causality. The old motto "correlation is not causation" applies vehemently here.
Remark. A funny anecdote about this expression "correlation is not causation". This expression is due to Karl Pearson, the father of modern statistics. He coined the phrase in his famous book "The Grammar of Science." Pearson is famous for inventing the correlation coefficient. He actually thought that correlation was a much superior, much more rigorous term, than causation. In his book, he actually used the sentence to argue in favor of abandoning causation altogether and focusing on the much better-defined and measurable concept of correlation. Interesting turn of events that his sentence is now used to mean that correlation is weaker than causation, totally reverting the original intended meaning.

In this section, we are going to define both comparisons, study their biases and state the conditions under which they identify TT. This will prove to be a very useful introduction to the notion of identification. It is

also very important to be able to understand the sources of bias of comparisons that we use every day and that come very naturally to policy makers and lay people.
Remark. In this section, we state the definitions and formulae in the population. This is for two reasons. First, it is simpler, and lighter in terms of notation. Second, it emphasizes that the problems with intuitive comparisons are independent of sampling noise. Most of the results stated here for the population extend to the sample, replacing the expectation operator by the average operator. I will nevertheless give examples in the sample, since it is so much simpler to compute. I will denote sample equivalents of population estimators with a hat.

1.4.1 With/Without comparison, selection bias and cross-sectional confounders

The with/without comparison (WW) is very intuitive: just compare the outcomes of the treated and untreated individuals in order to estimate the causal effect. This approach is nevertheless generally biased. We call the bias of WW selection bias (SB). Selection bias is due to unobserved confounders that are distributed differently in the treatment and control group and that generate differences in outcomes even in the absence of the treatment. In this section, I define the WW estimator, derives its bias, introduces the confounders and states conditions under which it is unbiased.

1.4.1.1 With/Without comparison

The with/without comparison (WW) is very intuitive: just compare the outcomes of the treated and untreated individuals in order to estimate the causal effect.
Definition 1.5 (With/without comparison)**.** The with/without comparison is the difference between the expected outcomes of the treated and the expected outcomes of the untreated:

$$\Delta^Y_{WW} = \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i|D_i = 0].$$

Example 1.9. In the population, WW can be computed using the traditional formula for the expectation of a truncated normal distribution:

$$\begin{aligned}
\Delta^y_{WW} &= \mathbb{E}[y_i|D_i = 1] - \mathbb{E}[y_i|D_i = 0] \\
&= \mathbb{E}[y^1_i|D_i = 1] - \mathbb{E}[y^0_i|D_i = 0] \\
&= \mathbb{E}[\alpha_i|D_i = 1] + \mathbb{E}[\mu_i + \rho U^B_i|\mu_i + U^B_i \leq \bar{y}] - \mathbb{E}[\mu_i + \rho U^B_i|\mu_i + U^B_i > \bar{y}] \\
&= \bar{\alpha} + \theta \left(\bar{\mu} - \frac{\sigma^2_\mu}{\sqrt{\sigma^2_\mu + \sigma^2_U}} \frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)} \right) - \frac{\sigma^2_\mu + \rho\sigma^2_U}{\sqrt{\sigma^2_\mu + \sigma^2_U}} \left(\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)} + \frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)}{1 - \Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)} \right).
\end{aligned}$$

In order to compute this parameter, we are going to set up a R function. For reasons that will become clearer later, we will define two separate functions to compute the first and second part of the formula. In the first part, you should have recognised TT, that we have already computed in Lecture 1. We are going to call the second part SB, for reasons that will become explicit in a bit.

```
delta.y.tt <- function(param){
  return(param["baralpha"]+param["theta"]*param["barmu"]-param["theta"]
         *((param["sigma2mu"]*dnorm((log(param["barY"])-param["barmu"])
                                    /(sqrt(param["sigma2mu"]+param["sigma2U"])))))
           /(sqrt(param["sigma2mu"]+param["sigma2U"])*pnorm((log(param["barY"])-param["barmu"])
                                                           /(sqrt(param["sigma2mu"]+param["sigma2U"])))
```

```
}
delta.y.sb <- function(param){
  return(-(param["sigma2mu"]+param["rho"]*param["sigma2U"])/sqrt(param["sigma2mu"]+param["sigma2U"])
         *dnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
         *(1/pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
           +1/(1-pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))))
}
delta.y.ww <- function(param){
  return(delta.y.tt(param)+delta.y.sb(param))
}
```

As a conclusion of all these derivations, WW in the population is equal to -1.298. Remember that the value of TT in the population is 0.172.

In order to compute the WW estimator in a sample, I'm going to generate a brand new sample and I'm going to choose a seed for the pseudo-random number generator so that we obtain the same result each time we run the code. I use `set.seed(1234)` in the code chunk below.

```
param <- c(8,.5,.28,1500)
names(param) <- c("barmu","sigma2mu","sigma2U","barY")
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
Ds[YB<=param["barY"]] <- 1
l <- length(param)
param <- c(param,0.9,0.01,0.05,0.05,0.05,0.1)
names(param)[(l+1):length(param)] <- c("rho","theta","sigma2epsilon","sigma2eta","delta","baralpha")
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In this sample, the average outcome of the treated in the presence of the treatment is

$$\frac{1}{\sum_{i=1}^{N} D_i} \sum_{i=1}^{N} D_i y_i = 7.084.$$

It is materialized by a circle on Figure 1.6. The average outcome of the untreated is

$$\frac{1}{\sum_{i=1}^{N} (1-D_i)} \sum_{i=1}^{N} (1-D_i) y_i = 8.378.$$

It is materialized by a plus sign on Figure 1.6.

The estimate of the WW comparison in the sample is thus:

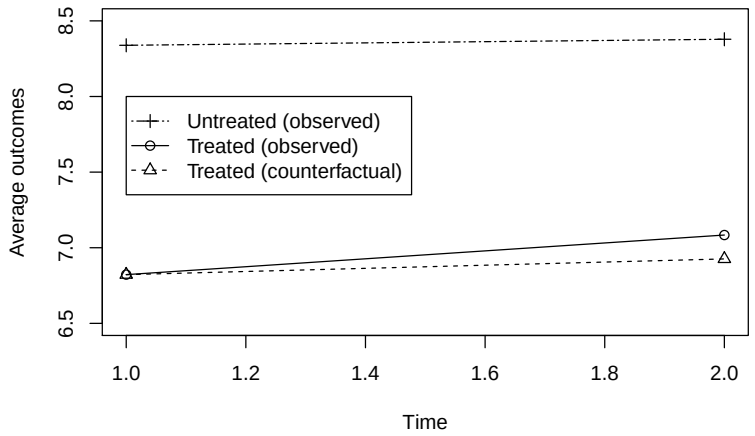


Figure 1.6: Evolution of average outcomes in the treated and control group before (Time =1) and after (Time=2) the treatment

$$\Delta^{\hat{Y}}_{WW} = \frac{1}{\sum_{i=1}^{N} D_i} \sum_{i=1}^{N} Y_i D_i - \frac{1}{\sum_{i=1}^{N} (1 - D_i)} \sum_{i=1}^{N} Y_i (1 - D_i).$$

We have $\Delta^{\hat{y}}_{WW}$ = -1.295. Remember that the value of TT in the sample is $\Delta^{y}_{TT_s} = 0.158$.

Overall, WW severely underestimates the effect of the treatment in our example. WW suggests that the treatment has a negative effect on outcomes whereas we know by construction that it has a positive one.

1.4.1.2 Selection bias

When we form the with/without comparison, we do not recover the TT parameter. Instead, we recover TT plus a bias term, called **selection bias**:

$$\Delta^{Y}_{WW} = \Delta^{Y}_{TT} + \Delta^{Y}_{SB}.$$

Definition 1.6 (Selection bias)**.** Selection bias is the difference between the with/without comparison and the treatment on the treated parameter:

$$\Delta^{Y}_{SB} = \Delta^{Y}_{WW} - \Delta^{Y}_{TT}.$$

WW tries to approximate the counterfactual expected outcome in the treated group by using $\mathbb{E}[Y^0_i | D_i = 0]$, the expected outcome in the untreated group . Selection bias appears because this proxy is generally poor. It is very easy to see that selection bias is indeed directly due to this bad proxy problem:
Theorem 1.2 (Selection bias and counterfactual)**.** *Selection bias is the difference between the counterfactual expected potential outcome in the absence of the treatment among the treated and the expected potential outcome in the absence of the treatment among the untreated.*

$$\Delta^{Y}_{SB} = \mathbb{E}[Y^0_i | D_i = 1] - \mathbb{E}[Y^0_i | D_i = 0].$$

Proof.

$$\begin{aligned}\Delta^Y_{SB} &= \Delta^Y_{WW} - \Delta^Y_{TT} \\ &= \mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i|D_i=0] - \mathbb{E}[Y_i^1 - Y_i^0|D_i=1] \\ &= \mathbb{E}[Y_i^0|D_i=1] - \mathbb{E}[Y_i^0|D_i=0].\end{aligned}$$

The first and second equalities stem only from the definition of both parameters. The third equality stems from using the switching equation: $Y_i = Y_i^1 D_i + Y_i^0(1-D_i)$, so that $\mathbb{E}[Y_i|D_i=1] = \mathbb{E}[Y_i^1|D_i=1]$ and $\mathbb{E}[Y_i|D_i=0] = \mathbb{E}[Y_i^0|D_i=0]$. □

Example 1.10. In the population, SB is equal to

$$\begin{aligned}\Delta^y_{SB} &= \Delta^y_{WW} - \Delta^y_{TT} \\ &= -1.298 - 0.172 \\ &= -1.471\end{aligned}$$

We could have computed SB directly using the formula from Theorem 1.2:

$$\begin{aligned}\Delta^y_{SB} &= \mathbb{E}[y_i^0|D_i=1] - \mathbb{E}[y_i^0|D_i=0] \\ &= -\frac{\sigma^2_\mu + \rho\sigma^2_U}{\sqrt{\sigma^2_\mu + \sigma^2_U}} \left(\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)} + \frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)}{1-\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma^2_\mu+\sigma^2_U}}\right)} \right).\end{aligned}$$

When using the R function for SB that we have defined earlier, we indeed find: Δ^y_{SB} = -1.471.

In the sample, $\Delta^{\hat{y}}_{SB}$ =-1.295-0.158 = -1.452. Selection bias emerges because we are using a bad proxy for the counterfactual. The average outcome for the untreated is equal to $\frac{1}{\sum_{i=1}^N (1-D_i)} \sum_{i=1}^N (1-D_i) y_i = 8.378$ while the counterfactual average outcome for the treated is $\frac{1}{\sum_{i=1}^N D_i} \sum_{i=1}^N D_i y_i^0 = 6.926$. Their difference is as expected equal to SB: $\Delta^{\hat{y}}_{SB} = 6.926 - 8.378$ = -1.452. The counterfactual average outcome of the treated is much smaller than the average outcome of the untreated. On Figure 1.6, this is materialized by the fact that the plus sign is located much above the triangle.

Remark. The concept of selection bias is related to but different from the concept of sample selection bias. With sample selection bias, we worry that selection into the sample might bias the estimated effect of a treatment on outcomes. With selection bias, we worry that selection into the treatment itself might bias the effect of the treatment on outcomes. Both biases are due to unbserved covariates, but they do not play out in the same way.

For example, estimating the effect of education on women's wages raises both selection bias and sample selection bias issues. Selection bias stems from the fact that more educated women are more likely to be more dynamic and thus to have higher earnings even when less educated. Selection bias would be positive in that case, overestimating the effect of education on earnings.

Sample selection bias stems from the fact that we can only use a sample of working women in order to estimate the effect of education on wages, since we do not observe the wages on non working women. But, selection into the labor force might generate sample selection bias. More educated women participate more in the labor market, while less educated women participate less. As a consequence, less educated women that work are different from the overall sample of less educated women. They might be more dynamic and work-focused. As a consequence, their wages are higher than the average wages of the less educated women.

Comparing the wages of less educated women that work to those of more educated women that work might understate the effect of education on earnings. Sample selection bias would generate a negative bias on the education coefficient.

1.4.1.3 Confounding factors

Confounding factors are the factors that generate differences between treated and untreated individuals even in the absence of the treatment. The confounding factors are thus responsible for selection bias. In general, the mere fact of being selected for receiving the treatment means that you have a host of characteristics that would differentiate you from the unselected individuals, even if you were not to receive the treatment eventually.

For example, if a drug is given to initially sicker individuals, then, we expect that they will be sicker that the untreated in the absence of the treatment. Comparing sick individuals to healthy ones is not a sound way to estimate the effect of a treatment. Obviously, even if our treatment performs well, healthier individuals will be healthier after the treatment has been allocated to the sicker patients. The best we can expect is that the treated patients have recovered, and that their health after the treatment is comparable to that of the untreated patients. In that case, the with/without comparison is going to be null, whereas the true effect of the treatment is positive. Selection bias is negative in that case: in the absence of the treatment, the average health status of the treated individuals would have been smaller than that of the untreated individuals. The confounding factor is the health status of individuals when the decision to allocate the drug has been taken. It is correlated to both the allocation of the treatment (negatively) and to health in the absence of the treatment (positively).

Example 1.11. In our example, μ_i and U_i^B are the confounding factors. Because the treatment is only given to individuals with pre-treament outcomes smaller than a threshold ($y_i^B \leq \bar{y}$), participants tend to have smaller μ_i and U_i^B than non participants, as we can see on Figure 1.7.

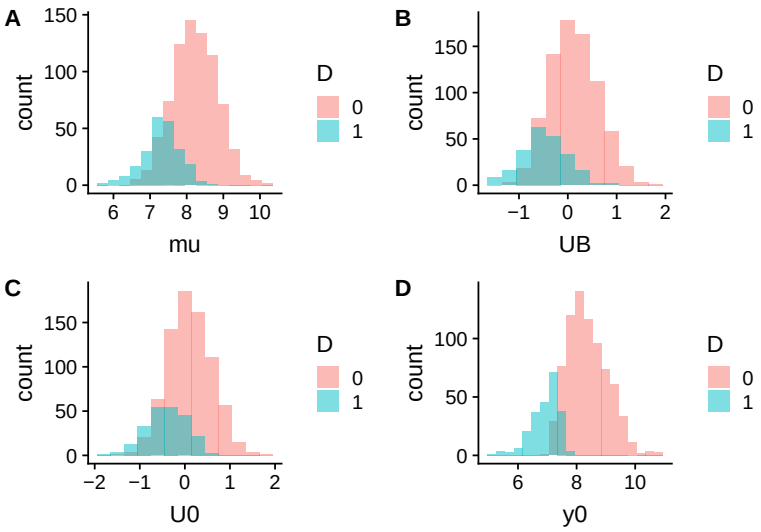


Figure 1.7: Distribution of confounders in the treated and control group

Since confounding factors are persistent, they affect the outcomes of participants and non participants after the treatment date. μ_i persists entirely over time, and U_i^B persists at a rate ρ. As a consequence, even in the absence of the treatment, participants have lower outcomes than non participants, as we can see on Figure 1.7.

We can derive the contributions of both confouding factors to overall SB:

$$
\begin{aligned}
\mathbb{E}[Y_i^0|D_i=1] &= \mathbb{E}[\mu_i+\delta+U_i^0|\mu_i+U_i^B\leq\bar{y}] \\
&= \delta+\mathbb{E}[\mu_i|\mu_i+U_i^B\leq\bar{y}]+\rho\mathbb{E}[U_i^B|\mu_i+U_i^B\leq\bar{y}] \\
\Delta_{SB}^y &= \mathbb{E}[\mu_i|\mu_i+U_i^B\leq\bar{y}]-\mathbb{E}[\mu_i|\mu_i+U_i^B>\bar{y}] \\
&\quad+\rho\left(\mathbb{E}[U_i^B|\mu_i+U_i^B\leq\bar{y}]-\mathbb{E}[U_i^B|\mu_i+U_i^B>\bar{y}]\right) \\
&= -\frac{\sigma_\mu^2}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\left(\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}+\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{1-\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}\right) \\
&\quad-\frac{\rho\sigma_U^2}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\left(\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}+\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{1-\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}\right)
\end{aligned}
$$

In order to evaluate these quantities, let's build two R functions:

```
delta.y.sb.mu <- function(param){
  return(-(param["sigma2mu"])/sqrt(param["sigma2mu"]+param["sigma2U"])
         *dnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
         *(1/pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
           +1/(1-pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))))
}
delta.y.sb.U <- function(param){
  return(-(param["rho"]*param["sigma2U"])/sqrt(param["sigma2mu"]+param["sigma2U"])
         *dnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
         *(1/pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
           +1/(1-pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))))
}
```

The contribution of μ_i to selection bias is -0.978 while that of U_i^0 is of -0.493.

1.4.1.4 When does WW identify TT?

Are there conditions under which WW identify TT? The answer is yes: when there is no selection bias, the proxy used by WW for the counterfactual quantity is actually valid. Formally, WW identifies TT when the following assumption holds:

Definition 1.7 (No selection bias)**.** We assume the following:

$$\mathbb{E}[Y_i^0|D_i=1]=\mathbb{E}[Y_i^0|D_i=0].$$

Under Assumption 1.7, the expected counterfactual outcome of the treated is equal to the expected potential outcome of the untreated in the absence of the treatment. This yields to the following result:

Theorem 1.3. *Under Assumption 1.7, WW identifies the TT parameter:*

$$\Delta_{WW}^Y=\Delta_{TT}^Y.$$

Proof.

$$\begin{aligned}\Delta^Y_{WW} &= \mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i|D_i=0] \\ &= \mathbb{E}[Y^1_i|D_i=1] - \mathbb{E}[Y^0_i|D_i=0] \\ &= \mathbb{E}[Y^1_i|D_i=1] - \mathbb{E}[Y^0_i|D_i=1] \\ &= \Delta^Y_{TT},\end{aligned}$$

where the second equation uses the switching equation and the third uses Assumption 1.7. □

So, under Assumption 1.7, the WW comparison actually identifies the TT parameter. We say that Assumption 1.7 is an **identification assumption**: it serves to identify the parameter of interest using observed data. The intuition for this result is simply that, under Assumption 1.7, there are no confounding factors and thus no selection bias. under Assumption 1.7, the factors that yield individuals to receive or not the treatment are mean-independent of the potential outcomes in the absence of the treatment. In this case, the expected outcome in the untreated group actually is a perfect proxy for the counterfactual expected outcome of the treated group.

Obviously, Assumption 1.7 is extremely unlikely to hold in real life. For Assumption 1.7 to hold, it has to be that **all** the determinants of D_i are actually unrelated to Y^0_i. One way to enforce Assumption 1.7 is to randomize treatment intake. We will see this in the Lecture on RCTs. It might also be possible that Assumption 1.7 holds in the data in the absence of an RCT. But this is not very likely, and should be checked by every mean possible.

One way to test for the validity of Assumption 1.7 is to compare the values of observed covariates in the treated and untreated group. For Assumption 1.7 to be credible, observed covariates should be distributed in the same way.

Another nice way to test for the validity of Assumption 1.7 with observed data is to implement a **placebo test**. A placebo test looks for an effect where there should be none, if we believe the identification assumptions. For example, under Assumption 1.7 it should be (even though it is not rigorously implied) that outcomes before the treatment are also mean-independent of the treatment allocation. And actually, since a future treatment cannot have an effect today (unless people anticipate the treatment, which we assume away here), the WW comparison before the treatment should be null, therefore giving a zero effect of the placebo treatment "will receive the treatment in the future."

Example 1.12. When the allocation rule defining D_i is the eligibility rule that we have used so far, we have already seen that Assumption 1.7 does not hold and the placebo test should not pass either.

One way of generating Assumption 1.7 from the eligibility rule that we are using is to mute the persistence in outcome dynamics. For example, one could set $\rho = 0$ and $\sigma^2_\mu = 0$.

```
param <- c(8,0,.28,1500,0,0.01,0.05,0.05,0.05,0.1)
names(param) <- c("barmu","sigma2mu","sigma2U","barY","rho","theta","sigma2epsilon","sigma2eta","delta"
param
```

```
##         barmu      sigma2mu       sigma2U          barY           rho
##          8.00          0.00          0.28       1500.00          0.00
##         theta sigma2epsilon     sigma2eta         delta      baralpha
##          0.01          0.05          0.05          0.05          0.10
```

In that case, outcomes are not persistent and Assumption 1.7 holds:

$$\begin{aligned}\mathbb{E}[y_i^0|D_i=1] &= \mathbb{E}[\mu_i+\delta+U_i^0|y_i^B\leq\bar{y}]\\ &= \mathbb{E}[\bar{\mu}+\delta+\epsilon_i|\bar{\mu}+U_i^B\leq\bar{y}]\\ &= \bar{\mu}+\delta+\mathbb{E}[\epsilon_i|\bar{\mu}+U_i^B\leq\bar{y}]\\ &= \bar{\mu}+\delta+\mathbb{E}[\epsilon_i|\bar{\mu}+U_i^B>\bar{y}]\\ &= \mathbb{E}[\mu_i+\delta+U_i^0|y_i^B>\bar{y}]\\ &= \mathbb{E}[y_i^0|D_i=0],\end{aligned}$$

where the second equality follows from $\sigma_\mu^2=0$ and $\rho=0$ and the fourth from $\epsilon_i \perp\!\!\!\perp U_i^B$. Another direct way to see this is to use the formula for selection bias that we have derived above. It is easy to see that with $\rho=0$ and $\sigma_\mu^2=0$, $\Delta_{SB}^y=0$. To be sure, we can compute Δ_{SB}^y with the new parameter values: $\Delta_{SB}^y=0$. As a consequence, $\Delta_{TT}^y=0.18=0.18=\Delta_{WW}^y$.

Remark. You might have noticed that the value of Δ_{TT}^y is different than before. It is normal, since it depends on the values of parameters, and especially on σ_μ^2 and ρ.

Let's see how these quantities behave in the sample.

```
set.seed(1234)
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
Ds[YB<=param["barY"]] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

We can see that $\mathbb{E}[Y_i^0|\hat{D}_i=1]=8.086\approx 8.038=\mathbb{E}[Y_i^0|\hat{D}_i=0]$. This means that WW should be close to TT: $\hat{\Delta_{TT}^y}=0.16\approx 0.208=\hat{\Delta_{WW}^y}$. Note that $\hat{WW}$ in the sample is not exactly, but only approximately, equal to TT in the population and in the sample. This is an instance of the Fundamental Problem of Statistical Inference that we will study in the next chapter.

Under these restrictions, the placebo test would unfortunately conclude against Assumption 1.7 even though it is valid:

$$\begin{aligned}\mathbb{E}[y_i^B|D_i=1] &= \mathbb{E}[\mu_i+U_i^B|y_i^B\leq\bar{y}]\\ &= \mathbb{E}[\bar{\mu}+U_i^B|\bar{\mu}+U_i^B\leq\bar{y}]\\ &= \bar{\mu}+\mathbb{E}[U_i^B|\bar{\mu}+U_i^B\leq\bar{y}]\\ &\neq \bar{\mu}+\mathbb{E}[U_i^B|\bar{\mu}+U_i^B>\bar{y}]\\ &= \mathbb{E}[\mu_i+U_i^0|y_i^B>\bar{y}]\\ &= \mathbb{E}[y_i^B|D_i=0].\end{aligned}$$

In the sample, we indeed have that $\mathbb{E}[Y_i^B|\hat{D}_i = 1] = 7.05 \neq 8.091 = \mathbb{E}[Y_i^B|\hat{D}_i = 0]$. The reason for the failure of the placebo test to conclude that Ww is actually correct is that the U_i^B shock enters both into the selection equation and the outcome equation for y_i^B, generating a wage at period B between the outcomes of the treated and of the untreated. Since it is not persistent, this wedge does not generate selection bias. This wedge would not be detected if we could perform it further back in time, before the selection period.

Another way to make Assumption 1.7 work is to generate a new allocation rule where all the determinants of treatment intake are indeed orthogonal to potential outcomes and to outcomes before the treatment. Let's assume for example that $D_i = \mathbb{1}[V_i \leq \bar{y}]$, with $V_i \sim \mathcal{N}(\bar{\mu}, \sigma_\mu^2 + \sigma_U^2)$ and $V_i \perp\!\!\!\perp (Y_i^0, Y_i^1, Y_i^B, \mu_i, \eta_i)$. In that case, Assumption 1.7 holds and the placebo test does work. Indeed, we have:

$$\begin{aligned}
\Delta_{TT}^y &= \mathbb{E}[Y_i^1 - Y_i^0|D_i = 1] \\
&= \mathbb{E}[\alpha_i|D_i = 1] \\
&= \mathbb{E}[\bar{\alpha} + \theta\mu_i + \eta_i|V_i \leq \bar{y}] \\
&= \bar{\alpha} + \theta\bar{\mu} \\
&= \Delta_{ATE}^y \\
\Delta_{WW}^y &= \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i|D_i = 0] \\
&= \mathbb{E}[Y_i^1|D_i = 1] - \mathbb{E}[Y_i^0|D_i = 0] \\
&= \mathbb{E}[Y_i^1|V_i \leq \bar{y}] - \mathbb{E}[Y_i^0|V_i > \bar{y}] \\
&= \mathbb{E}[Y_i^1] - \mathbb{E}[Y_i^0] \\
&= \Delta_{ATE}^y
\end{aligned}$$

ATE is the Average Treatment Effect in the population. It is the expected effect of the treatment on all the members of the population, not only on the treated. When the treatment is randomly allocated, both TT and ATE are equal, since the treated are a random subset of the overall population. I prefer to use ATE for my definition of the R function in order not to erase the definition of the TT function:

```
delta.y.ate <- function(param){
  return(param["baralpha"]+param["theta"]*param["barmu"])
}
```

In the population, WW identifies TT: $\Delta_{TT}^y = 0.18 = \Delta_{WW}^y$. Let's see how these quantities behave in the sample:

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
```

```
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In the sample, the counterfactual is well approximated by the outcomes of the untreated: $\mathbb{E}[Y_i^0|\hat{D}_i = 1] = 8.038 \approx 8.055 = \mathbb{E}[Y_i^0|\hat{D}_i = 0]$. As a consequence, WW should be close to TT: $\Delta^{\hat{y}}_{TT} = 0.183 \approx 0.167 = \Delta^{\hat{y}}_{WW}$. The placebo test is also valid in that case: $\mathbb{E}[Y_i^B|\hat{D}_i = 1] = 7.922 \approx 8.016 = \mathbb{E}[Y_i^B|\hat{D}_i = 0]$.

1.4.2 The before/after comparison, temporal confounders and time trend bias

The before/after comparison (BA) is also very intuitive: it consists in looking at how the outcomes of the treated have changed over time and to attribute this change to the effect of the treatment. The problem is that other changes might have affected outcomes in the absence of the treatment, thereby biasing BA. The bias of BA is called time-trend bias. It is due to confounders that affect the outcomes of the treated over time. This section defines the BA estimator, derives its bias, describes the role of the confounders and states conditions under which BA identifies TT.

Example 1.13. Before computing any estimates, we need to reset all our parameter values and generated sample it their usual values:

```
param <- c(8,.5,.28,1500)
names(param) <- c("barmu","sigma2mu","sigma2U","barY")
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
Ds[YB<=param["barY"]] <- 1
l <- length(param)
param <- c(param,0.9,0.01,0.05,0.05,0.05,0.1)
names(param)[(l+1):length(param)] <- c("rho","theta","sigma2epsilon","sigma2eta","delta","baralpha")
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

1.4.2.1 The before/after comparison

The before/after estimator (BA) compares the outcomes of the treated after taking the treatment to the outcomes of the treated before taking the treatment. It is also sometimes called a "pre-post comparison."

Definition 1.8 (Before/after comparison)**.** The before/after comparison is the difference between the expected outcomes in the treated group after the treatment and the expected outcomes in the same group before the treatment:

$$\Delta^Y_{BA} = \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i^B|D_i = 1].$$

Example 1.14. In the population, the BA estimator has the following shape:

$$\begin{aligned}\Delta^y_{BA} &= \mathbb{E}[y_i|D_i=1] - \mathbb{E}[y_i^B|D_i=1] \\ &= \mathbb{E}[y_i^1 - y_i^B|D_i=1] \\ &= \mathbb{E}[\alpha_i|D_i=1] + \delta + (\rho-1)\mathbb{E}[U_i^B|\mu_i + U_i^B \leq \bar{y}] \\ &= \Delta^y_{TT} + \delta + (1-\rho)\left(\frac{\sigma_U^2}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}\right).\end{aligned}$$

In order to compute BA in the population, we can again use a R function, combining the value of TT and that of the second part of the formula, that we are going to denote TB for reasons that are going to become clear in a bit.

```
delta.y.tb <- function(param){
  return(param["delta"]
          +(1-param["rho"])*((param["sigma2U"])/sqrt(param["sigma2mu"]+param["sigma2U"]))
         *dnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"])))
         /pnorm((log(param["barY"])-param["barmu"])/(sqrt(param["sigma2mu"]+param["sigma2U"]))))
}
delta.y.ba <- function(param){
  return(delta.y.tt(param)+ delta.y.tb(param))
}
```

The value of BA in the population is thus $\Delta^y_{BA} = 0.265$. Remember that the true value of TT in the population is 0.172. In the sample, the value of BA is $\Delta^{\hat{y}}_{BA} = 0.262$. Remember that the value of TT in the sample is $\Delta^y_{TT_s} = 0.158$.

1.4.2.2 Time trend bias

When we form the before/after comparison, we do not recover the TT parameter. Instead, we recover TT plus a bias term, called **time trend bias**:

$$\Delta^Y_{BA} = \Delta^Y_{TT} + \Delta^Y_{TB}.$$

Definition 1.9 (Time trend bias)**.** Time trend bias is the difference between the before/after comparison and the treatment on the treated parameter:

$$\Delta^Y_{TB} = \Delta^Y_{BA} - \Delta^Y_{TT}.$$

BA uses the expected outcome in the treated group before the treatment as a proxy for the expected counterfactual outcome in the absence of the treatment in the same group. TB is due to the fact that BA uses an imperfect proxy for the counterfactual expected outcome of the treated:

Theorem 1.4. *Time trend bias is the difference between the counterfactual expected potential outcome in the absence of the treatment among the treated and the expected outcome before the treatment in the same group.*

$$\Delta^Y_{TB} = \mathbb{E}[Y_i^0|D_i=1] - \mathbb{E}[Y_i^B|D_i=1].$$

Proof.

$$\begin{aligned}\Delta^Y_{TB} &= \Delta^Y_{BA} - \Delta^Y_{TT} \\ &= \mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i^B|D_i=1] - \mathbb{E}[Y_i^1 - Y_i^0|D_i=1] \\ &= \mathbb{E}[Y_i^0|D_i=1] - \mathbb{E}[Y_i^B|D_i=1].\end{aligned}$$

The first and second equalities stem from the definition of both parameters. The third equality stems from using the switching equation: $Y_i = Y_i^1 D_i + Y_i^0(1-D_i)$, so that $\mathbb{E}[Y_i|D_i=1] = \mathbb{E}[Y_i^1|D_i=1]$. □

Example 1.15. In the population, TB is equal to

$\Delta^y_{TB} = \Delta^y_{BA} - \Delta^y_{TT} = 0.265 - 0.172 = 0.093$. We could have computed this result using Theorem1.4:

$$\begin{aligned}\Delta^y_{TB} &= \mathbb{E}[y_i^0|D_i=1] - \mathbb{E}[y_i^B|D_i=1] \\ &= \delta + (1-\rho)\left(\frac{\sigma_U^2}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\frac{\phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}{\Phi\left(\frac{\bar{y}-\bar{\mu}}{\sqrt{\sigma_\mu^2+\sigma_U^2}}\right)}\right).\end{aligned}$$

Using the R function that we have defined previously, this approach gives $\Delta^y_{TB} = 0.093$.

In the sample $\Delta^{\hat{y}}_{BA} = 0.262$ while $\Delta^{\hat{y}}_{TT} = 0.158$, so that $\Delta^{\hat{y}}_{TB} = 0.104$. Time trend bias emerges because we are using a bad proxy for the counterfactual average outcomes of the treated. The average outcome of the treated before the treatment takes place is $\mathbb{E}[y_i^B|\hat{D}_i=1] = 6.822$ while the true counterfactual average outcome for the treated after the treatment is $\mathbb{E}[y_i^0|\hat{D}_i=1] = 6.926$. Outcomes would have increased in the treatment group even in the absence of the treatment. As a consequence, the BA comparison overestimates the true effect of the treatment. TB estimated using Theorem 1.4 is equal to: $\Delta^{\hat{y}}_{TB} = 6.926 - 6.822 = 0.104$. This can be seen on Figure 1.6: the triangle in period 2 is higher than in period 1.

1.4.2.3 Temporal confounders

Temporal confounders make the outcomes of the treated change at the same time as the treatment status changes, thereby confounding the effect of the treatment. Temporal confounders are responsible for time trend bias.

Over time, there are other things that change than the treatment status. For example, maybe sick individuals naturally recover, and thus their counterfactual health status is better than ther health status before taking the treatment. As a results, BA might overstimate the effect of the treatment. It might also be that the overall level of health in the country has increased, because of increasing GDP, for example.
Example 1.16. In our example, δ and U_i^B are the confounders. δ captures the overall changes in outcomes over time (business cycle, general improvement of health status). U_i^B captures the fact that transitorily sicker individuals tend at the same time to receive the treatment and also to recover naturally. The BA comparison incorrectly attributes both of these changes to the effect of the treatment. We can compute the relative contributions of both sources to the overall time-trend bias in the population.

δ contributes for 0.05 while U_i^B contributes for 0.043.

1.4.2.4 When does BA identify TT?

Are there conditions under which BA actually identifies TT? The answer is yes, when there are no temporal confounders. When that is the case, the variation of outcomes over time is only due to the treatment and it identifies the treatment effect.

More formally, we make the following assumption:
Definition 1.10 (No time trend bias)**.** We assume the following:

$$\mathbb{E}[Y_i^0|D_i = 1] = \mathbb{E}[Y_i^B|D_i = 1].$$

Under Assumption 1.10, the expected counterfactual outcome of the treated is equal to the expected potential outcome of the untreated in the absence of the treatment. This yields to the following result:
Theorem 1.5. *Under Assumption 1.10, BA identifies the TT parameter:*

$$\Delta_{BA}^Y = \Delta_{TT}^Y.$$

Proof.

$$\begin{aligned}\Delta_{BA}^Y &= \mathbb{E}[Y_i|D_i = 1] - \mathbb{E}[Y_i^B|D_i = 1] \\ &= \mathbb{E}[Y_i^1|D_i = 1] - \mathbb{E}[Y_i^B|D_i = 1] \\ &= \mathbb{E}[Y_i^1|D_i = 1] - \mathbb{E}[Y_i^0|D_i = 1] \\ &= \Delta_{TT}^Y,\end{aligned}$$

where the second equation uses the switching equation and the third uses Assumption 1.10. □

Under Assumption 1.10 the outcomes of the treated before the treatment takes place are a good proxy for the counterfactual. As a consequence, BA identifies TT. Under Assumption 1.10is very unlikely to hold in real life. Indeed, it requires tha nothing happens to the outcomes of the treated in the absence of the treatment. Assumption 1.10 rules out economy-wide shocks but also mean-reversion, such as when sick people naturally recover from an illness.

A good way to test for the validity of Assumption 1.10 is to perform a placebo test. Two of these tests are possible. One placebo test would be to apply the BA estimator between two pre-treatment periods where nothing should happen since the treatment status does not vary and, by assumption, nothing else should vary. A second placebo test would be to apply the BA estimator to a group that does not receive the treatment. The untreated group is a perfectly suitable candidate for that. Assumption 1.10 does not imply that there should be no change in the untreated outcomes, but detecting such a change would cast a serious doubt on the validity of Assumption 1.10.
Example 1.17. One way to generate a population in which Assumption 1.10 holds is to shut down the two sources of confounders in our original model by setting both $\delta = 0$ and $\rho = 1$.

```
param <- c(8,0.5,.28,1500,1,0.01,0.05,0.05,0,0.1)
names(param) <- c("barmu","sigma2mu","sigma2U","barY","rho","theta","sigma2epsilon","sigma2eta","delta"
param
```

```
##         barmu      sigma2mu       sigma2U          barY           rho
##          8.00          0.50          0.28       1500.00          1.00
##         theta sigma2epsilon     sigma2eta         delta      baralpha
##          0.01          0.05          0.05          0.00          0.10
```

In that case, according to the formula we have derived for TB, we have: $\Delta_{TB}^y = 0$. Let's see how these quantities behave in the sample:

```
set.seed(1234)
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
```

```
YB <- exp(yB)
Ds <- rep(0,N)
Ds[YB<=param["barY"]] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In the sample, the value of BA is $\Delta^{\hat{y}}_{BA} = 0.164$ while the value of TT in the sample is $\Delta^{y}_{TT_s} = 0.158$. We cannot perform a placebo test using two periods of pre-treatment outcomes for the treated since we have generated only one period of pre-treatment outcome. We will be able to perform this test later in the DID lecture. We can perfom the placebo test that applies the BA estimator to the untreated. The value of BA for the untreated is $\hat{\Delta^{y}_{BA|D=0}} = 0.002$, which is reasonably close to zero.

Chapter 2

Fundamental Problem of Statistical Inference

The Fundamental Problem of Statistical Inference (FPSI) states that, even if we have an estimator E that identifies TT in the population, we cannot observe E because we only have access to a finite sample of the population. The only thing that we can form from the sample is a sample equivalent $\hat{E}$ to the population quantity E, and $\hat{E} \neq E$. For example, the sample analog to WW is the difference in means between treated and untreated units $\hat{WW}$. As we saw in the last lecture, $\hat{WW}$ is never exactly equal to WW.

Why is $\hat{E} \neq E$? Because a finite sample is never perfectly representative of the population. In a sample, even in a random sample, the distribution of the observed and unobserved covariates deviates from the true population one. As a consequence, the sample value of the estimator is never precisely equal to the population value, but fluctuates around it with sampling noise. The main problem with the FPSI is that if we find an effect of our treatment, be it small or large, we cannot know whether we should attribute it to the treatment or to the bad or good luck of sampling noise.

What can we do to deal with the FPSI? I am going to argue that there are mainly two things that we might want to do: estimating the extent of sampling noise and decreasing sampling noise.

Estimating sampling noise means measuring how much variability there is in our estimate $\hat{E}$ due to the sampling procedure. This is very useful because it enables us to form a confidence interval that gauges how far from $\hat{E}$ the true value E might be. It is a measure of the precision of our estimation and of the extent to which sampling noise might drive our results. Estimating sampling noise is very hard because we have only access to one sample and we would like to know the behavior of our estimator over repeated samples. We are going to learn four ways to estimate the extent of sampling noise using data from one sample.

Because sampling noise is such a nuisance and makes our estimates imprecise, we would like to be able to make it as small as possible. We are going to study three ways of decreasing sampling noise, two that take place before collecting the data (increasing sample size, stratifying) and one that takes place after (conditioning).

Maybe you are surprised not to find statistical significance tests as an important answer to the FPSI. I argue in this lecture that statistical tests are misleading tools that make us overestimate the confidence in our results and underestimate the scope of sampling noise. Statistical tests are not meant to be used for scientific research, but were originally designed to make decisions in industrial settings where the concept of successive sampling made actual sense. Statistical tests also generate collective behaviors such as publication bias and specification search that undermine the very foundations of science. A general movement in the social sciences, but also in physics, is starting to ban the reporting of p-values.

2.1 What is sampling noise? Definition and illustration

In this section, I am going to define sampling noise and illustrate it with a numerical example. In Section 2.1.1, I define sampling noise. In section 2.1.2, I illustrate how sampling noise varies when one is interested in the population treatment effect. In section 2.1.3, I illustrate how sampling noise varies when one is interesetd in the sample treatment effect. Finally, in section 2.1.4, I show how confidence intervals can be built from an estimate of sampling noise.

2.1.1 Sampling noise, a definition

Sampling noise measures how much sampling variability moves the sample estimator $\hat{E}$ around. One way to define it more rigorously is to make it equal to the width of a confidence interval:
Definition 2.1 (Sampling noise)**.** Sampling noise is the width of the symmetric interval around TT within which $\delta * 100\%$ of the sample estimators fall, where δ is the confidence level. As a consequence, sampling noise is equal to 2ϵ where ϵ is such that:

$$\Pr(|\hat{E} - TT| \leq \epsilon) = \delta.$$

This definition tries to capture the properties of the distribution of $\hat{E}$ using only one number. As every simplification, it leaves room for dissatisfaction, exactly as a 2D map is a convenient albeit arbitrary betrayal of a 3D phenomenon. For example, there is nothing sacred about the symmetry of the interval. It is just extremely convenient. One might prefer an interval that is symmetric in tail probabilities instead. Feel free to explore with different concepts if you like.

A related concept to that of sampling noise is that of precision: the smaller the sampling noise, the higher the precision. Precision can be defined for example as the inverse of sampling noise $\frac{1}{2\epsilon}$.

Finally, a very useful concept is that of signal to noise ratio. It is not used in economics, but physicists use this concept all the time. The signal to noise ratio measures the treatment effect in multiple of the sampling noise. If they are of the same order of magnitude, we have a lot of noise and little confidence in our estimates. If the signal is much larger than the noise, we tend to have a lot of confidence in our parameter estimates. The signal to noise ratio can be computed as follows: $\frac{E}{2\epsilon}$ or $\frac{\hat{E}}{2\epsilon}$.
Remark. A very striking result is that the signal to noise ratio of a result that is marginally significant at the 5% level is very small, around one half, meaning that the noise is generally double the signal in these results. We will derive this result after studying how to estimate sampling noise with real data.

There are two distinct ways of understanding sampling noise, depending on whether we are after the population treatment effect (Δ^Y_{TT}) or the sample treatment effect ($\Delta^Y_{TT_s}$). Sampling noise for the population treatment effect stems from the fact that the sample is not perfectly representative of the population. The sample differs from the population and thus the sample estimates differs from the population estimate. Sampling noise for the sample parameter stems from the fact that the control group is not a perfect embodiment of the counterfactual. Discrepancies between treated and control samples are going to generate differences between the WW estimate and the TT effect in the sample.

2.1.2 Sampling noise for the population treatment effect

Sampling noise for the population treatment effect stems from the fact that the sample is not perfectly representative of the population.
Example 2.1. In order to assess the scope of sampling noise for our population treatment effect estimate, let's first draw a sample. In order to be able to do that, I first have to define the parameter values:

```
param <- c(8,.5,.28,1500,0.9,0.01,0.05,0.05,0.05,0.1)
names(param) <- c("barmu","sigma2mu","sigma2U","barY","rho","theta","sigma2epsilon","sigma2eta","delta"
param
```

```
##         barmu      sigma2mu       sigma2U          barY           rho
##          8.00          0.50          0.28       1500.00          0.90
##         theta sigma2epsilon     sigma2eta         delta      baralpha
##          0.01          0.05          0.05          0.05          0.10
```

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)

delta.y.ate <- function(param){
  return(param["baralpha"]+param["theta"]*param["barmu"])
}
```

In this sample, the WW estimator yields an estimate of $\hat{\Delta^y_{WW}} = 0.133$. Despite random assignment, we have $\hat{\Delta^y_{WW}} \neq \Delta^y_{TT} = 0.18$, an instance of the FPSI.

In order to see how sampling noise varies, let's draw another sample. In order to do so, I am going to choose a different seed to initialize the pseudo-random number generator in R.

```
set.seed(12345)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
```

```
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In this sample, the WW estimator yields an estimate of $\Delta^{\hat{y}}_{WW} = 0.179$. Again, despite random assignment, we have $\Delta^{\hat{y}}_{WW} \neq \Delta^{y}_{TT} = 0.18$, an instance of the FPSI. Furthermore, the estimate of the population treatment effect in this sample differs from the previous one, a consequence of sampling noise.

Let's now visualize the extent of sampling noise by repeating the procedure multiple times with various sample sizes. This is called Monte Carlo replications: in each replication, I choose a sample size, draw one sample from the population and compute the $\hat{WW}$ estimator. At each replication, the sample I'm using is different, reflecting the actual sampling process and enabling me to gauge the extent of sampling noise. In order to focus on sampling noise alone, I am running the replications in the model in which selection into the treatment is independent on potential outcomes, so that $WW = TT$ in the population. In order to speed up the process, I am using parallelized computing: I send each sample to a different core in my computer so that several samples can be run at the same time. You might want to adapt the program below to the number of cores you actually have using the `ncpus` variable in the beginning of the `.Rmd` file that generates this page.. In order to parallelize computations, I use the Snowfall package in R, that gives very simple and intuitive parallelization commands. In order to save time when generating the graph, I use the wonderful "cache" option of knitr: it stores the estimates from the code chunk and will not rerun it as long as the code inside the chunk has not been altered nor the code of the chunks that it depends on (parameter values, for example).

```
monte.carlo.ww <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  Ds <- rep(0,N)
  V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
  Ds[V<=log(param["barY"])] <- 1
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  return(c((1/sum(Ds))*sum(y*Ds)-(1/sum(1-Ds))*sum(y*(1-Ds)),var(y[Ds==1]),var(y[Ds==0]),mean(Ds)))
}

simuls.ww.N <- function(N,Nsim,param){
  simuls.ww <- as.data.frame(matrix(unlist(lapply(1:Nsim,monte.carlo.ww,N=N,param=param)),nrow=Nsim,nco
  colnames(simuls.ww) <- c('WW','V1','V0','p')
  return(simuls.ww)
}

sf.simuls.ww.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=ncpus)
  sim <- as.data.frame(matrix(unlist(sfLapply(1:Nsim,monte.carlo.ww,N=N,param=param)),nrow=Nsim,ncol=4,
```

```
  sfStop()
  colnames(sim) <- c('WW','V1','V0','p')
  return(sim)
}

simuls.ww <- lapply(N.sample,sf.simuls.ww.N,Nsim=Nsim,param=param)

par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww[[i]][,'WW'],main=paste('N=',as.character(N.sample[i])),xlab=expression(hat(Delta^yWW))
  abline(v=delta.y.ate(param),col="red")
}
```

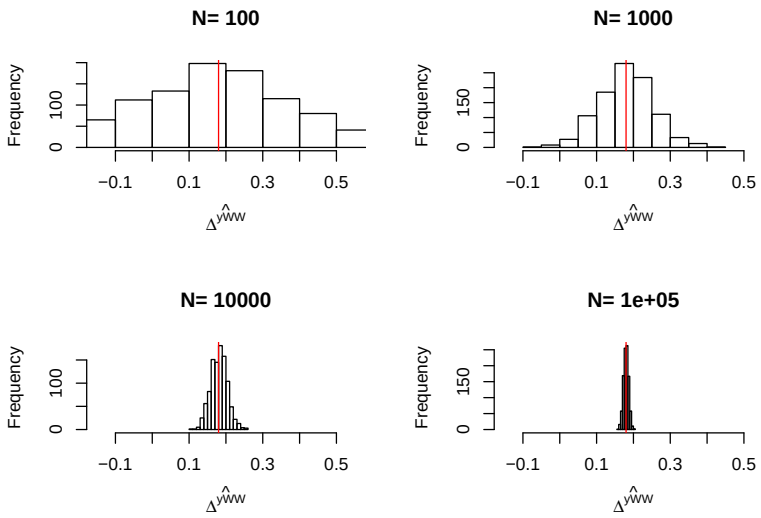


Figure 2.1: Distribution of the WW estimator over replications of samples of different sizes

Figure 2.1 is essential to understanding statistical inference and the properties of our estimators. We can see on Figure 2.1 that the estimates indeed move around at each sample replication. We can also see that the estimates seem to be concentrated around the truth. We also see that the estimates are more and more concentrated around the truth as sample size grows larger and larger.

How big is sampling noise in all of these examples? We can compute it by using the replications as approximations to the true distribution of the estimator after an infinite number of samples has been drawn. Let's first choose a confidence level and then compute the empirical equivalent to the formula in Definition 2.1.

```
delta<- 0.99
delta.2 <- 0.95
samp.noise <- function(estim,delta){
  return(2*quantile(abs(delta.y.ate(param)-estim),prob=delta))
}
samp.noise.ww <- sapply(lapply(simuls.ww,`[`,,1),samp.noise,delta=delta)
names(samp.noise.ww) <- N.sample
samp.noise.ww
```

```
##        100       1000      10000      1e+05
## 1.09916429 0.39083801 0.11582492 0.03527744
```

Let's also compute precision and the signal to noise ratio and put all of these results together in a nice table.

Table 2.1: Sampling noise of $\hat{WW}$ for the population treatment effect with $\delta = 0.99$ for various sample sizes

	Sampling noise	Precision	Signal to noise ratio
100	1.10	0.91	0.16
1000	0.39	2.56	0.46
10000	0.12	8.63	1.55
1e+05	0.04	28.35	5.10

```
precision <- function(estim,delta){
  return(1/samp.noise(estim,delta))
}
signal.to.noise <- function(estim,delta,param){
  return(delta.y.ate(param)/samp.noise(estim,delta))
}
precision.ww <- sapply(lapply(simuls.ww,`[`,,1),precision,delta=delta)
names(precision.ww) <- N.sample
signal.to.noise.ww <- sapply(lapply(simuls.ww,`[`,,1),signal.to.noise,delta=delta,param=param)
names(signal.to.noise.ww) <- N.sample
table.noise <- cbind(samp.noise.ww,precision.ww,signal.to.noise.ww)
colnames(table.noise) <- c('Sampling noise', 'Precision', 'Signal to noise ratio')
knitr::kable(table.noise,caption=paste('Sampling noise of $\\hat{WW}$ for the population treatment effe
```

Finally, a nice way to summarize the extent of sampling noise is to graph how sampling noise varies around the true treatment effect, as shown on Figure 2.2.

```
colnames(table.noise) <- c('sampling.noise', 'precision', 'signal.to.noise')
table.noise <- as.data.frame(table.noise)
table.noise$N <- as.numeric(rownames(table.noise))
table.noise$TT <- rep(delta.y.ate(param),nrow(table.noise))
ggplot(table.noise, aes(x=as.factor(N), y=TT)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dod
  xlab("Sample Size")+
  theme_bw()
```

With $N = 100$, we can definitely see on Figure 2.2 that sampling noise is ridiculously large, especially compared with the treatment effect that we are trying to estimate. The signal to noise ratio is 0.16, which means that sampling noise is an order of magnitude bigger than the signal we are trying to extract. As a consequence, in 22.2% of our samples, we are going to estimate a negative effect of the treatment. There is also a 20.4% chance that we end up estimating an effect that is double the true effect. So how much can we trust our estimate from one sample to be close to the true effect of the treatment when $N = 100$? Not much.

With $N = 1000$, sampling noise is still large: the signal to noise ratio is 0.46, which means that sampling noise is double the signal we are trying to extract. As a consequence, the chance that we end up with a negative treatment effect has decreased to 0.9% and that we end up with an effect double the true one is 1%. But still, the chances that we end up with an effect that is smaller than three quarters of the true effect is 25.6% and the chances that we end up with an estimator that is 25% bigger than the true effect is 26.2%. These are nontrivial differences: compare a program that increases earnings by 13.5% to one that increases them by 18% and another by 22.5%. They would have completely different cost/benefit ratios. But we at least trust our estimator to give us a correct idea of the sign of the treatment effect and a vague and imprecise idea of its magnitude.

With $N = 10^{4}$, sampling noise is smaller than the signal, which is encouraging. The signal to noise ratio is 1.55. In only 1% of the samples does the estimated effect of the treatment become smaller than 0.125

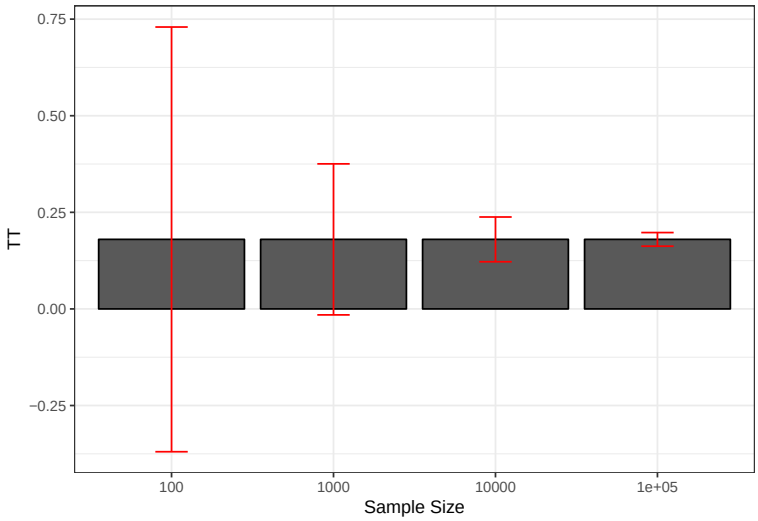


Figure 2.2: Sampling noise of $\hat{WW}$ (99% confidence) around TT for various sample sizes

or bigger than 0.247. We start gaining a lot of confidence in the relative magnitude of the effect, even if sampling noise is still responsible for economically significant variation.

With $N = 10\hat{}\{5\}$, sampling noise has become trivial. The signal to noise ratio is 5.1, which means that the signal is now 5 times bigger than the sampling noise. In only 1% of the samples does the estimated effect of the treatment become smaller than 0.163 or bigger than 0.198. Sampling noise is not any more responsible for economically meaningful variation.

2.1.3 Sampling noise for the sample treatment effect

Sampling noise for the sample parameter stems from the fact that the treated and control groups are not perfectly identical. The distribution of observed and unobserved covariates is actually different, because of sampling variation. This makes the actual comparison of means in the sample a noisy estimate of the true comparison that we would obtain by comparing the potential outcomes of the treated directly.

In order to understand this issue well and to be able to illustrate it correctly, I am going to focus on the average treatment effect in the whole sample, not on the treated: $\Delta^{Y}_{ATE_s} = \frac{1}{N}\sum_{i=1}^{N}(Y_i^1 - Y_i^0)$. This enables me to define a sample parameter that is independent of the allocation of D_i. This is without important consequences since these two parameters are equal in the population when there is no selection bias, as we are assuming since the beginning of this lecture. Furthermore, if we view the treatment allocation generating no selection bias as a true random assignment in a Randomized Controlled Trial (RCT), then it is still possible to use this approach to estimate TT if we view the population over which we randomise as the population selected for receiving the treatment, as we will see in the lecture on RCTs.

Example 2.2. In order to assess the scope of sampling noise for our sample treatment effect estimate, we first have to draw a sample:

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
```

```
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In this sample, the treatment effect parameter is $\Delta^y_{ATE_s} = 0.171$. The WW estimator yields an estimate of $\Delta^{\hat{y}}_{WW} = 0.133$. Despite random assignment, we have $\Delta^y_{ATE_s} \neq \Delta^{\hat{y}}_{WW}$, an instance of the FPSI.

In order to see how sampling noise varies, let's draw a new treatment allocation, while retaining the same sample and the same potential outcomes.

```
set.seed(12345)
N <-1000
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In this sample, the treatment effect parameter is still $\Delta^y_{ATE_s} = 0.171$. The WW estimator yields now an estimate of $\Delta^{\hat{y}}_{WW} = 0.051$. The WW estimate is different from our previous estimate because the treatment was allocated to a different random subset of people.

Why is this second estimate so imprecise? It might because it estimates one of the two components of the average treatment effect badly, or both. The true average potential outcome with the treatment is, in this sample, $\frac{1}{N}\sum_{i=1}^{N} y_i^1 = 8.207$ while the WW estimate of this quantity is $\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i y_i = 8.113$. The true average potential outcome without the treatment is, in this sample, $\frac{1}{N}\sum_{i=1}^{N} y_i^0 = 8.036$ while the WW estimate of this quantity is $\frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}(1-D_i)y_i = 8.062$. It thus seems that most of the bias in the estimated effect stems from the fact that the treatment has been allocated to individuals with lower than expected outcomes with the treatment, be it because they did not react strongly to the treatment, or because they were in worse shape without the treatment. We can check which one of these two explanations is more important. The true average effect of the treatment is, in this sample, $\frac{1}{N}\sum_{i=1}^{N}(y_i^1 - y_i^0) = 0.171$ while, in the treated group, this quantity is $\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i(y_i^1 - y_i^0) = 0.18$. The true average potential outcome without the treatment is, in this sample, $\frac{1}{N}\sum_{i=1}^{N} y_i^0 = 8.036$ while, in the treated group, this quantity is $\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i y_i^0 = 7.933$. The reason for the poor performance of the WW estimator in this sample is that individuals with lower counterfactual outcomes were included in the treated group, not that the treatment had lower effects on them. The bad counterfactual outcomes of the treated generates a bias of -0.103, while the bias due to heterogeneous reactions to the treatment is of 0.009. The last part of the bias is the one due to the fact that the individuals in the control group have slightly better counterfactual outcomes than in the sample: -0.026. The sum of these three terms yields the total bias of our WW estimator in this second sample: -0.12.

Let's now assess the overall effect of sampling noise on the estimate of the sample treatment effect for various sample sizes. In order to do this, I am going to use parallelized Monte Carlo simulations again. For the sake of simplicity, I am going to generate the same potential outcomes in each replication, using the same seed, and only choose a different treatment allocation.

```
monte.carlo.ww.sample <- function(s,N,param){
  set.seed(1234)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  set.seed(s)
  Ds <- rep(0,N)
  V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
  Ds[V<=log(param["barY"])] <- 1
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  return((1/sum(Ds))*sum(y*Ds)-(1/sum(1-Ds))*sum(y*(1-Ds)))
}

simuls.ww.N.sample <- function(N,Nsim,param){
  return(unlist(lapply(1:Nsim,monte.carlo.ww.sample,N=N,param=param)))
}

sf.simuls.ww.N.sample <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=ncpus)
  sim <- sfLapply(1:Nsim,monte.carlo.ww.sample,N=N,param=param)
  sfStop()
  return(unlist(sim))
}

simuls.ww.sample <- lapply(N.sample,sf.simuls.ww.N.sample,Nsim=Nsim,param=param)

monte.carlo.ate.sample <- function(N,s,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  Ds <- rep(0,N)
  V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
  Ds[V<=log(param["barY"])] <- 1
```

```
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  return(mean(alpha))
}

par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww.sample[[i]],main=paste('N=',as.character(N.sample[i])),xlab=expression(hat(Delta^yWW))
  abline(v=monte.carlo.ate.sample(N.sample[[i]],1234,param),col="red")
}
```

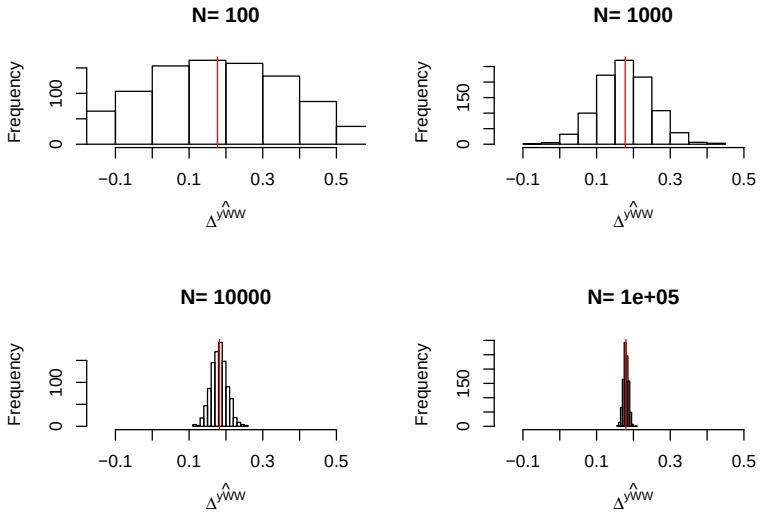


Figure 2.3: Distribution of the WW estimator over replications of treatment allocation for samples of different sizes

Let's also compute sampling noise, precision and the signal to noise ratio in these examples.

```
samp.noise.sample <- function(i,delta,param){
  return(2*quantile(abs(monte.carlo.ate.sample(1234,N.sample[[i]],param)-simuls.ww.sample[[i]]),prob=del
}
samp.noise.ww.sample <- sapply(1:4,samp.noise.sample,delta=delta,param=param)
names(samp.noise.ww.sample) <- N.sample

precision.sample <- function(i,delta,param){
  return(1/samp.noise.sample(i,delta,param=param))
}
signal.to.noise.sample <- function(i,delta,param){
  return(monte.carlo.ate.sample(1234,N.sample[[i]],param)/samp.noise.sample(i,delta,param=param))
}
precision.ww.sample <- sapply(1:4,precision.sample,delta=delta,param=param)
names(precision.ww.sample) <- N.sample
signal.to.noise.ww.sample <- sapply(1:4,signal.to.noise.sample,delta=delta,param=param)
names(signal.to.noise.ww.sample) <- N.sample
table.noise.sample <- cbind(samp.noise.ww.sample,precision.ww.sample,signal.to.noise.ww.sample)
colnames(table.noise.sample) <- c('Sampling noise', 'Precision', 'Signal to noise ratio')
knitr::kable(table.noise.sample,caption=paste('Sampling noise of $\\hat{WW}$ for the sample treatment e
```

Finally, let's compare the extent of sampling noise for the population and the sample treatment effect parameters.

Table 2.2: Sampling noise of $\hat{WW}$ for the sample treatment effect with $\delta = 0.99$ and for various sample sizes

	Sampling noise	Precision	Signal to noise ratio
100	1.208	0.828	0.149
1000	0.366	2.729	0.482
10000	0.122	8.218	1.585
1e+05	0.033	30.283	5.453

```
colnames(table.noise.sample) <- c('sampling.noise', 'precision', 'signal.to.noise')
table.noise.sample <- as.data.frame(table.noise.sample)
table.noise.sample$N <- as.numeric(rownames(table.noise.sample))
table.noise.sample$TT <- sapply(N.sample,monte.carlo.ate.sample,s=1234,param=param)
table.noise.sample$Type <- 'TTs'
table.noise$Type <- 'TT'
table.noise.tot <- rbind(table.noise,table.noise.sample)
table.noise.tot$Type <- factor(table.noise.tot$Type)

ggplot(table.noise.tot, aes(x=as.factor(N), y=TT,fill=Type)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dod
  xlab("Sample Size")+
  theme_bw()+
  theme(legend.position=c(0.85,0.88))
```

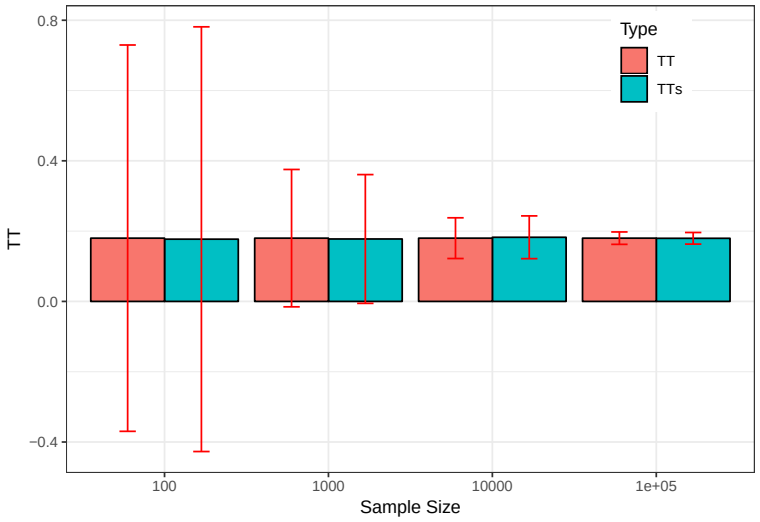


Figure 2.4: Sampling noise of $\hat{WW}$ (99% confidence) around TT and TT_s for various sample sizes

Figure 2.3 and Table 2.2 present the results of the simulations of sampling noise for the sample treatment effect parameter. Figure 2.4 compares sampling noise for the population and sample treatment effects.
For all practical purposes, the estimates of sampling noise for the sample treatment effect are extremely close to the ones we have estimated for the population treatment effect. I am actually surprised by this result, since I expected that keeping the potential outcomes constant over replications would decrease sampling noise. It seems that the variability in potential outcomes over replications of random allocations of the treatment in a given sample mimicks very well the sampling process from a population. I do not know if this result of similarity of sampling noise for the population and sample treatment effect is a general one, but considering them as similar or close seems innocuous in our example.

2.1.4 Building confidence intervals from estimates of sampling noise

In real life, we do not observe TT. We only have access to $\hat{E}$. Let's also assume for now that we have access to an estimate of sampling noise, 2ϵ. How can we use these two quantities to assess the set of values that TT might take? One very useful device that we can use is the confidence interval. Confidence intervals are very useful because they quantify the zone within which we have a chance to find the true effect TT:

Theorem 2.1 (Confidence interval)**.** *For a given level of confidence δ and corresponding level of sampling noise 2ϵ of the estimator $\hat{E}$ of TT, the confidence interval $\left\{\hat{E}-\epsilon, \hat{E}+\epsilon\right\}$ is such that the probability that it contains TT is equal to δ over sample replications:*

$$\Pr(\hat{E}-\epsilon \leq TT \leq \hat{E}+\epsilon) = \delta.$$

Proof. From the definition of sampling noise, we know that:

$$\Pr(|\hat{E}-TT| \leq \epsilon) = \delta.$$

Now:

$$\begin{aligned}\Pr(|\hat{E}-TT| \leq \epsilon) &= \Pr(TT-\epsilon \leq \hat{E} \leq TT+\epsilon)\\ &= \Pr(-\hat{E}-\epsilon \leq -TT \leq -\hat{E}+\epsilon)\\ &= \Pr(\hat{E}-\epsilon \leq TT \leq \hat{E}+\epsilon),\end{aligned}$$

which proves the result. □

It is very important to note that confidence intervals are centered around $\hat{E}$ and not around TT. When estimating sampling noise and building Figure 2.2, we have centered our intervals around TT. The interval was fixed and $\hat{E}$ was moving across replications and 2ϵ was defined as the length of the interval around TT containing a proportion δ of the estimates $\hat{E}$. A confidence interval cannot be centered around TT, which is unknown, but is centered around $\hat{E}$, that we can observe. As a consequence, it is the interval that moves around across replications, and δ is the proportion of samples in which the interval contains TT.

Example 2.3. Let's see how confidence intervals behave in our numerical example.

```
N.plot <- 40
plot.list <- list()

for (k in 1:length(N.sample)){
  set.seed(1234)
  test <- sample(simuls.ww[[k]][,'WW'],N.plot)
  test <- as.data.frame(cbind(test,rep(samp.noise(simuls.ww[[k]][,'WW'],delta=delta)),rep(samp.noise(si
  colnames(test) <- c('WW','sampling.noise.1','sampling.noise.2')
  test$id <- 1:N.plot
  plot.test <- ggplot(test, aes(x=as.factor(id), y=WW)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=WW-sampling.noise.1/2, ymax=WW+sampling.noise.1/2), width=.2,position=posi
      geom_errorbar(aes(ymin=WW-sampling.noise.2/2, ymax=WW+sampling.noise.2/2), width=.2,position=posi
      geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
      #ylim(-0.5,1.2)+
      xlab("Sample id")+
      theme_bw()+
```

```
      ggtitle(paste("N=",N.sample[k]))
  plot.list[[k]] <- plot.test
}
plot.CI <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(N.s
print(plot.CI)
```

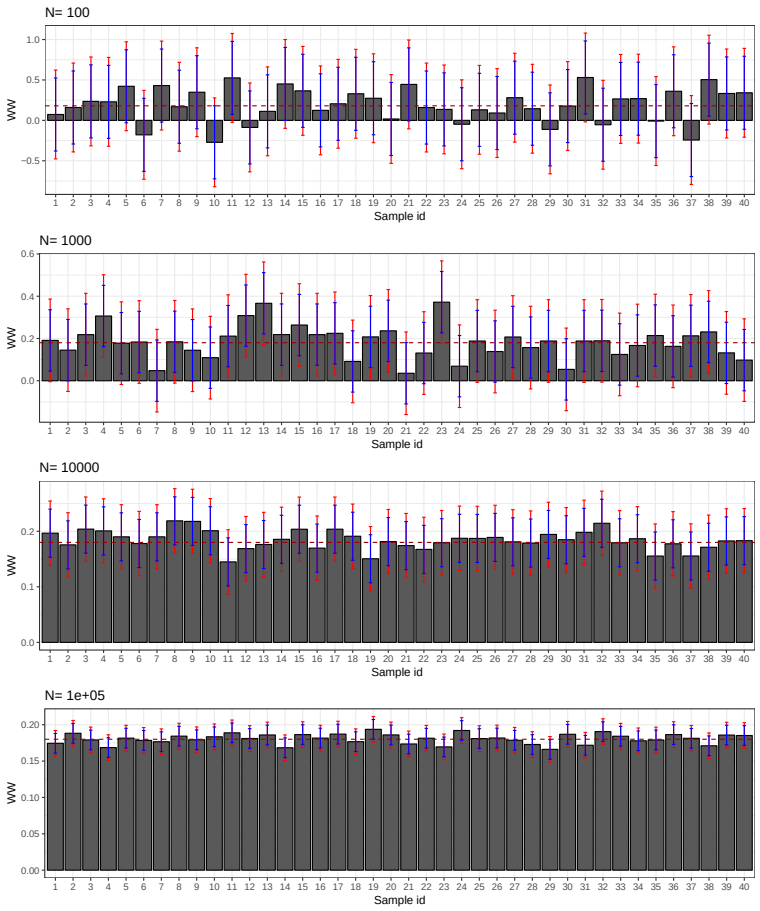


Figure 2.5: Confidence intervals of $W\hat{W}$ for $\delta = 0.99$ (red) and 0.95 (blue) over sample replications for various sample sizes

Figure 2.5 presents the 99% and 95% confidence intervals for 40 samples selected from our simulations. First, confidence intervals do their job: they contain the true effect most of the time. Second, the 95% confidence interval misses the true effect more often, as expected. For example, with $N = 1000$, the confidence intervals in samples 13 and 23 do not contain the true effect, but it is not far from their lower bound. Third, confidence intervals faithfully reflect what we can learn from our estimates at each sample size. With $N = 100$, the confidence intervals make it clear that the effect might be very large or very small, even strongly negative. With $N = 1000$, the confidence intervals suggest that the effect is either positive or null, but unlikely to be strongly negative. Most of the time, we get the sign right. With $N = 10\hat{}\{4\}$, we know that the true effect is bigger than 0.1 and smaller than 0.3 and most intervals place the true effect somewhere between 0.11 and 0.25. With $N = 10\hat{}\{5\}$, we know that the true effect is bigger than 0.15 and smaller than 0.21 and most intervals place the true effect somewhere between 0.16 and 0.20.

2.1.5 Reporting sampling noise: a proposal

Once sampling noise is measured (and we'll see how to get an estimate in the next section), one still has to communicate it to others. There are many ways to report sampling noise:

- Sampling noise as defined in this book ($2 * \epsilon$)
- The corresponding confidence interval
- The signal to noise ratio
- A standard error
- A significance level
- A p-value
- A t-statistic

The main problem with all of these approaches is that they do not express sampling noise in a way that is directly comparable to the magnitude of the TT estimate. Other ways of reporting sampling noise such as p-values and t-stats are nonlinear transforms of sampling noise, making it difficult to really gauge the size of sampling noise as it relates to the magnitude of TT.

My own preference goes to the following format for reporting results: $TT \pm \epsilon$. As such, we can readily compare the size of the noise to the sizee of the TT estimate. We can also form all the other ways of expressing sampling noise directly.

Example 2.4. Let's see how this approach behaves in our numerical example.

```
test.all <- list()
for (k in 1:length(N.sample)){
  set.seed(1234)
  test <- sample(simuls.ww[[k]][,'WW'],N.plot)
  test <- as.data.frame(cbind(test,rep(samp.noise(simuls.ww[[k]][,'WW'],delta=delta)),rep(samp.noise(si
  colnames(test) <- c('WW','sampling.noise.1','sampling.noise.2')
  test$id <- 1:N.plot
  test.all[[k]] <- test
}
```

With $N = 100$, the reporting of the results for sample 1 would be something like: "we find an effect of 0.07 ± 0.55." Note how the choice of δ does not matter much for the result. The previous result was for $\delta = 0.99$ while the result for $\delta = 0.95$ would have been: "we find an effect of 0.07 ± 0.45." The precise result changes with δ, but the qualitative result stays the same: the magnitude of sampling noise is large and it dwarfs the treatment effect estimate.

With $N = 1000$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect of 0.19 ± 0.2." With $\delta = 0.95$: "we find an effect of 0.19 ± 0.15." Again, although the precise quantitative result is affected by the choice of δ, but hte qualitative message stays the same: sampling noise is of the same order of magnitude as the estimated treatment effect.

With $N = 10\hat{}\{4\}$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect of 0.2 ± 0.06." With $\delta = 0.95$: "we find an effect of 0.2 ± 0.04." Again, see how the qualitative result is independent of the precise choice of δ: sampling noise is almost one order of magnitude smaller than the treatment effect estimate.

With $N = 10\hat{}\{5\}$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect of 0.17 ± 0.02." With $\delta = 0.95$: "we find an effect of 0.17 ± 0.01." Again, see how the qualitative result is independent of the precise choice of δ: sampling noise is one order of magnitude smaller than the treatment effect estimate.

Remark. What I hope the example makes clear is that my proposed way of reporting results gives the same importance to sampling noise as it gives to the treatment effect estimate. Also, comparing them is easy, without requiring a huge computational burden on our brain.

Remark. One problem with the approach that I propose is when you have a non-symetric distribution of sampling noise, or when $TT \pm \epsilon$ exceeds natural bounds on TT (such as if the effect cannot be bigger than

one, for example). I think these issues are minor and rare and can be dealt with on a case by case basis. The advantage of having one simple and directly readable number comparable to the magnitude of the treatment effect is overwhelming and makes this approach the most natural and adequate, in my opinion.

2.1.6 Using effect sizes to normalize the reporting of treatment effects and their precision

When looking at the effect of a program on an outcome, we depend on the scaling on that outcome to appreciate the relative size of the estimated treatment effect.
It is often difficult to appreciate the relative importance of the size of an effect, even if we know the scale of the outcome of interest. One useful device to normalize the treatment effects is called Cohen's d, or effect size. The idea is to compare the magnitude of the treatment effect to an estimate of the usual amount of variation that the outcome undergoes in the population. The way to build Cohen's d is by dividing the estimated treatment effect by the standard deviation of the outcome. I generally prefer to use the standard devaition of the outcome in the control group, so as not to include the additional amoiunt of variation due to the heterogeneity in treatment effects.
Definition 2.2 (Cohen's d)**.** Cohen's d, or effect size, is the ratio of the estimated treatment effect to the standard deviation of outcomes in the control group:

$$d = \frac{\hat{TT}}{\sqrt{\frac{1}{N^0}\sum_{i=1}^{N^0}(Y_i - \bar{Y^0})^2}}$$

where $\hat{TT}$ is an estimate of the treatment effect, N^0 is the number of individuals in the treatment group and $\bar{Y^0}$ is the average outcome in the treatment group.

Cohen's d can be interpreted in terms of magnitude of effect size:

- It is generally considered that an effect is large when its d is larger than 0.8.
- An effect size around 0.5 is considered medium
- An effect size around 0.2 is considered to be small
- An effect size around 0.02 is considered to be very small.

There probably could be a rescaling of these terms, but that is the actual state of the art.

What I like about effect sizes is that they encourage an interpretation of the order of magnitude of the treatment effect. As such, they enable to include the information on precision by looking at which orders of magnitude are compatible with the estimated effect at the estimated precision level. Effect sizes and orders of magnitude help make us aware that our results might be imprecise, and that the precise value that we have estimated is probably not the truth. What is important is the range of effect sizes compatible with our results (both point estimate and precision).
Example 2.5. Let's see how Cohen's d behaves in our numerical example.

The value of Cohen's d (or effect size) in the population is equal to:

$$ES = \frac{TT}{\sqrt{V^0}} = \frac{\bar{\alpha} + \theta\bar{\mu}}{\sqrt{\sigma_\mu^2 + \rho^2\sigma_U^2 + \sigma_\epsilon^2}}$$

We can write a function to compute this parameter, as well as functions to implement its estimator in the simulated samples:

```
V0 <- function(param){
  return(param["sigma2mu"]+param["rho"]^2*param["sigma2U"]+param["sigma2epsilon"])
}
```

```
ES <- function(param){
  return(delta.y.ate(param)/sqrt(V0(param)))
}

samp.noise.ES <- function(estim,delta,param=param){
  return(2*quantile(abs(delta.y.ate(param)/sqrt(V0(param))-estim),prob=delta))
}

for (i in 1:4){
  simuls.ww[[i]][,'ES'] <- simuls.ww[[i]][,'WW']/sqrt(simuls.ww[[i]][,'V0'])
}
```

The true effect size in the population is thus 0.2. It is considered to be small according to the current classification, although I'd say that a treatment able to move the outcomes by 20% of their usual variation is a pretty effective treatment, and this effect should be labelled at least medium. Let's stick with the classification though. In our example, the effect size does not differ much from the treatment effect since the standard deviation of outcomes in the control group is pretty close to one: it is equal to 0.88. Let's now build confidence intervals for the effect size and try to comment on the magnitudes of these effects using the normalized classification.

```
N.plot.ES <- 40
plot.list.ES <- list()

for (k in 1:length(N.sample)){
  set.seed(1234)
  test.ES <- sample(simuls.ww[[k]][,'ES'],N.plot)
  test.ES <- as.data.frame(cbind(test.ES,rep(samp.noise.ES(simuls.ww[[k]][,'ES'],delta=delta,param=para
  colnames(test.ES) <- c('ES','sampling.noise.ES.1','sampling.noise.ES.2')
  test.ES$id <- 1:N.plot.ES
  plot.test.ES <- ggplot(test.ES, aes(x=as.factor(id), y=ES)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=ES-sampling.noise.ES.1/2, ymax=ES+sampling.noise.ES.1/2), width=.2,positio
      geom_errorbar(aes(ymin=ES-sampling.noise.ES.2/2, ymax=ES+sampling.noise.ES.2/2), width=.2,positio
      geom_hline(aes(yintercept=ES(param)), colour="#990000", linetype="dashed")+
      #ylim(-0.5,1.2)+
      xlab("Sample id")+
      ylab("Effect Size")+
      theme_bw()+
      ggtitle(paste("N=",N.sample[k]))
  plot.list.ES[[k]] <- plot.test.ES
}
plot.CI.ES <- plot_grid(plot.list.ES[[1]],plot.list.ES[[2]],plot.list.ES[[3]],plot.list.ES[[4]],ncol=1,
print(plot.CI.ES)
```

Figure 2.6 presents the 99% and 95% confidence intervals for the effect size estimated in 40 samples selected from our simulations. Let's regroup our estimate and see how we could present their results.

```
test.all.ES <- list()
for (k in 1:length(N.sample)){
  set.seed(1234)
  test.ES <- sample(simuls.ww[[k]][,'ES'],N.plot)
  test.ES <- as.data.frame(cbind(test.ES,rep(samp.noise.ES(simuls.ww[[k]][,'ES'],delta=delta,param=para
  colnames(test.ES) <- c('ES','sampling.noise.ES.1','sampling.noise.ES.2')
  test.ES$id <- 1:N.plot.ES
  test.all.ES[[k]] <- test.ES
```

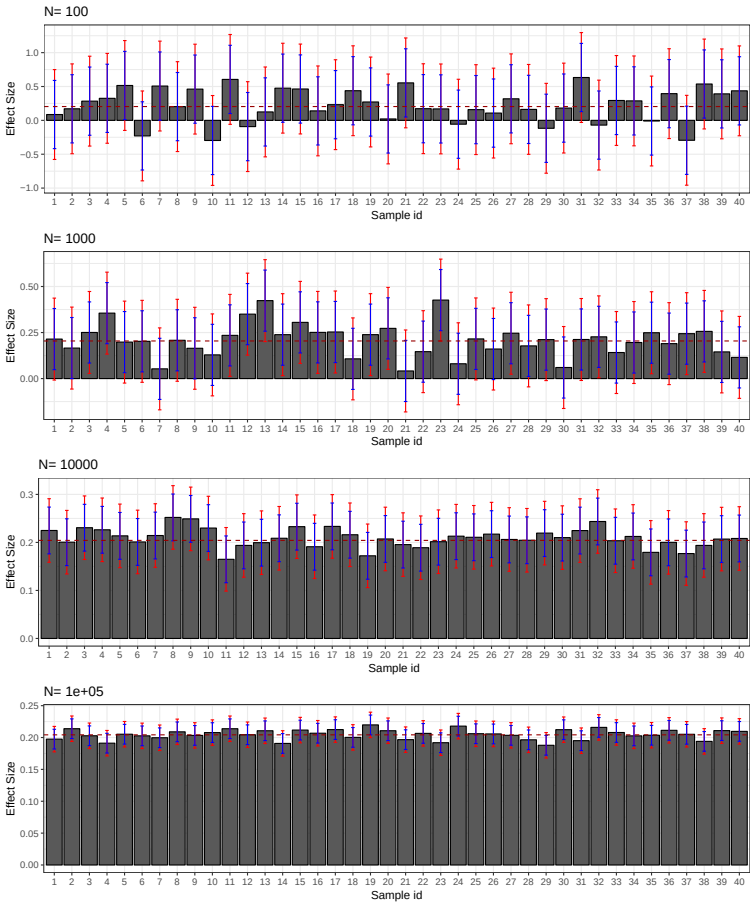


Figure 2.6: Confidence intervals of $\hat{ES}$ for $\delta = 0.99$ (red) and 0.95 (blue) over sample replications for various sample sizes

```
}
```

With $N = 100$, the reporting of the results for sample 1 would be something like: "we find an effect size of 0.09 ± 0.66" with $\delta = 0,99$. With $\delta = 0.95$ we would say: "we find an effect of 0.09 ± 0.5." All in all, our estimate is compatible with the treatment having a large positive effect size and a medium negative effect size. Low precision prevents us from saying much else.

With $N = 1000$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect size of 0.21 ± 0.22." With $\delta = 0.95$: "we find an effect size of 0.21 ± 0.17." Our estimate is compatible with a medium positive effect or a very small positive or even negative effect (depending on the choice of δ).

With $N = 10\hat{}\{4\}$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect size of 0.22 ± 0.07." With $\delta = 0.95$: "we find an effect size of 0.22 ± 0.05." Our estimate is thus compatible with a small effect of the treatment. We can rule out that the effect of the treatment is medium since the upper bound of the 99% confidence interval is equal to 0.29. We can also rule out that the effect of the treatment is very small since the lower bound of the 99% confidence interval is equal to 0.16. With this sample size, we have been able to reach a precision level sufficient enough to pin down the order of magnitude of the effect size of our treatment. There still remains a considerable amount of uncertainty about the true effect size, though: the upper bound of our confidence interval is almost double the lower bound.

With $N = 10\hat{}\{5\}$, the reporting of the results for sample 1 with $\delta = 0.99$ would be something like: "we find an effect size of 0.2 ± 0.02." With $\delta = 0.95$: "we find an effect size of 0.2 ± 0.02." Here, the level of precision of our result is such that, first, it does not depend on the choice of δ in any meaningful way, and second, we can do more than pinpoint the order of magnitude of the effect size, we can start to zero in on its precise value. From our estimate, the true value of the effect size is really close to 0.2. It could be equal to 0.18 or 0.22, but not further away from 0.2 than that. Remember that is actually equal to 0.2.

Remark. One issue with Cohen's d is that its magnitude depends on the dispersion of the outcomes in the control group. That means that for the same treatment, and same value of the treatment effect, the effect size is larger in a population where oucomes are more homogeneous. This is not an attractive feature of a normalizing scale that its size depends on the particular application. One solution would be, for each outcome, to provide a standardized scale, using for example the estmated standard deviation in a reference population. This would be similar to the invention of the metric system, where a reference scale was agreed uppon once and for all.

Remark. Cohen's d is well defined for continuous outcomes. For discrete outcomes, the use of Cohen's d poses a series of problems, and alternatives such as relative risk ratios and odds ratios have been proposed. I'll comment on that in the last chapter.

2.2 Estimating sampling noise

Gauging the extent sampling noise is very useful in order to be able to determine how much we should trust our results. Are they precise, so that the true treatment effect lies very close to our estimate? Or are our results imprecise, the true treatment effect maybe lying very far from our estimate?

Estimating sampling noise is hard because we want to infer a property of our estimator over repeated samples using only one sample. In this lecture, I am going to introduce four tools that enable you to gauge sampling noise and to choose sample size. The four tools are Chebyshev's inequality, the Central Limit Theorem, resampling methods and Fisher's permutation method. The idea of all these methods is to use the properties of the sample to infer the properties of our estimator over replications. Chebyshev's inequality gives an upper bound on the sampling noise and a lower bound on sample size, but these bounds are generally too wide to be useful. The Central Limit Theorem (CLT) approximates the distribution of $\hat{E}$ by a normal distribution, and quantifies sampling noise as a multiple of the standard deviation. Resampling methods use the sample as a population and draw new samples from it in order to approximate sampling noise. Fisher's permutation method, also called randomization inference, derives the distribution of $\hat{E}$ under the assumption that all treatment effects are null, by reallocating the treatment indicator among the treatment and control group.

Both the CLT and resampling methods are approximation methods, and their approximation of the true extent of sampling noise gets better and better as sample size increases. Fisher's permutation method is exact-it is not an approximation-but it only works for the special case of the WW estimator in a randomized design.

The remaining of this section is structured as follows. Section 2.2.1 introduces the assumptions that we will need in order to implement the methods. Section 2.2.2 presents the Chebyshev approach to gauging sampling noise and choosing sample size. Section 2.2.3 introduces the CLT way of approximating sampling noise and choosing sample size. Section 2.2.4 presents the resampling methods.
Remark. I am going to derive the estimators for the precision only for the WW estimator. In the following lectures, I will show how these methods adapt to other estimators.

2.2.1 Assumptions

In order to be able to use the theorems that power up the methods that we are going to use to gauge sampling noise, we need to make some assumptions on the properties of the data. The main assumptions that we need are that the estimator identifies the true effect of the treatment in the population, that the estimator is well-defined in the sample, that the observations in the sample are independently and identically distributed (i.i.d.), that there is no interaction between units and that the variances of the outcomes in the treated and untreated group are finite.

We know from last lecture that for the WW estimator to identify TT, we need to assume that there is no selection bias, as stated in Assumption 1.7. One way to ensure that this assumption holds is to use a RCT.

In order to be able to form the WW estimator in the sample, we also need that there is at least one treated and one untreated in the sample:
Definition 2.3 (Full rank)**.** We assume that there is at least one observation in the sample that receives the treatment and one observation that does not receive it:

$$\exists i, j \leq N \text{ such that } D_i = 1 \& D_j = 0.$$

One way to ensure that this assumption holds is to sample treated and untreated units.

In order to be able to estimate the variance of the estimator easily, we assume that the observations come from random sampling and are i.i.d.:
Definition 2.4 (i.i.d. sampling)**.** We assume that the observations in the sample are identically and independently distributed:

$$\begin{aligned} \forall i, j \leq N,\, i \neq j,\, (Y_i, D_i) &\perp\!\!\!\perp (Y_j, D_j), \\ (Y_i, D_i) \& (Y_j, D_j) &\sim F_{Y,D}. \end{aligned}$$

We have to assume something on how the observations are related to each other and to the population. Identical sampling is natural in the sense that we are OK to assume that the observations stem from the same population model. Independent sampling is something else altogether. Independence means that the fates of two closely related individuals are assumed to be independent. This rules out two empirically relevant scenarios:

1. The fates of individuals are related because of common influences, as for example the environment, etc,
2. The fates of individuals are related because they directly influence each other, as for example on a market, but also for example because there are diffusion effects, such as contagion of deseases or technological adoption by imitation.

We will address both sources of failure of the independence assumption in future lectures.

Finally, in order for all our derivations to make sense, we need to assume that the outcomes in both groups have finite variances, otherwise sampling noise is going to be too extreme to be able to estimate it using the methods developed in this lecture:

Definition 2.5 (Finite variance of $\Delta^{\hat{Y}}_{WW}$)**.** We assume that $\mathbb{V}[Y^1|D_i=1]$ and $\mathbb{V}[Y^0|D_i=0]$ are finite.

2.2.2 Using Chebyshev's inequality

Chebyshev's inequality is a fundamental building block of statistics. It relates the sampling noise of an estimator to its variance. More precisely, it derives an upper bound on the samplig noise of an unbiased estimator:

Theorem 2.2 (Chebyshev's inequality)**.** *For any unbiased estimator $\hat{\theta}$, sampling noise level 2ϵ and confidence level δ, sampling noise is bounded from above:*

$$2\epsilon \leq 2\sqrt{\frac{\mathbb{V}[\hat{\theta}]}{1-\delta}}.$$

Remark. The more general version of Chebyshev's inequality that is generally presented is as follows:

$$\Pr(|\hat{\theta} - \mathbb{E}[\hat{\theta}]| > \epsilon) \leq \frac{\mathbb{V}[\hat{\theta}]}{\epsilon^2}.$$

The version I present in Theorem 2.2 is adapted to the bouding of sampling noise for a given confidence level, while this version is adapted to bounding the confidence level for a given level of sampling noise. In order to go from this general version to Theorem 2.2, simply remember that, for an unbiased estimator, $\mathbb{E}[\hat{\theta}] = \theta$ and that, by definition of sampling noise, $\Pr(|\hat{\theta} - \theta| > \epsilon) = 1 - \delta$. As a result, $1 - \delta \leq \mathbb{V}[\hat{\theta}]/\epsilon^2$, hence the result in Theorem 2.2.

Using Chebyshev's inequality, we can obtain an upper bound on the sampling noise of the WW estimator:

Theorem 2.3 (Upper bound on the sampling noise of $\hat{WW}$)**.** *Under Assumptions 1.7, 2.3 and 2.4, for a given confidence level δ, the sampling noise of the $\hat{WW}$ estimator is bounded from above:*

$$2\epsilon \leq 2\sqrt{\frac{1}{N(1-\delta)}\left(\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}\right)} \equiv 2\bar{\epsilon}.$$

Proof. See in Appendix A.1.1 □

Theorem 2.3 is a useful step forward for estimating sampling noise. Theorem 2.3 states that the actual level of sampling noise of the $\hat{WW}$ estimator (2ϵ) is never bigger than a quantity that depends on sample size, confidence level and on the variances of outcomes in the treated and control groups. We either know all the components of the formula for $2\bar{\epsilon}$ or we can estimate them in the sample. For example, $\Pr(D_i = 1)$, $\mathbb{V}[Y_i^1|D_i=1]$ and $\mathbb{V}[Y_i^0|D_i=0]$ by can be approximated by, respectively:

$$\Pr(\hat{D_i} = 1) = \frac{1}{N}\sum_{i=1}^{N} D_i$$

$$\mathbb{V}[Y_i^1|\hat{D}_i = 1] = \frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i(Y_i - \frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i Y_i)^2$$

$$\mathbb{V}[Y_i^0|\hat{D}_i = 0] = \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}(1-D_i)(Y_i - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}(1-D_i)Y_i)^2.$$

Using these approximations for the quantities in the formula, we can compute an estimate of the upper bound on sampling noise, $2\hat{\bar{\epsilon}}$.

Example 2.6. Let's write an R function that is going to compute an estimate for the upper bound of sampling noise for any sample:

```
samp.noise.ww.cheb <- function(N,delta,v1,v0,p){
  return(2*sqrt((v1/p+v0/(1-p))/(N*(1-delta))))
}
```

Let's estimate this upper bound in our usual sample:

```
set.seed(1234)
N <-1000
delta <- 0.99
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
Ds[V<=log(param["barY"])] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

In our sample, for $\delta = 0.99$, $2\hat{\bar{\epsilon}} = 1.35$. How does this compare with the true extent of sampling noise when $N = 1000$? Remember that we have computed an estimate of sampling noise out of our Monte Carlo replications. In Table 2.2, we can read that sampling noise is actually equal to 0.39. The Chebyshev upper bound overestimates the extent of sampling noise by 245%.

How does the Chebyshev upper bound fares overall? In order to know that, let's compute the Chebyshev upper bound for all the simulated samples. You might have noticed that, when running the Monte Carlo simulations for the population parameter, I have not only recovered $\hat{WW}$ for each sample, but also the estimates of the components of the formula for the upper bound on sampling noise. I can thus easily compute the Chebyshev upper bound on sampling noise for each replication.

```
for (k in (1:length(N.sample))){
  simuls.ww[[k]]$cheb.noise <- samp.noise.ww.cheb(N.sample[[k]],delta,simuls.ww[[k]][,'V1'],simuls.ww[[
}
```

```
par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww[[i]][,'cheb.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expression(hat(2*ba
  abline(v=table.noise[i,colnames(table.noise)=='sampling.noise'],col="red")
}
```

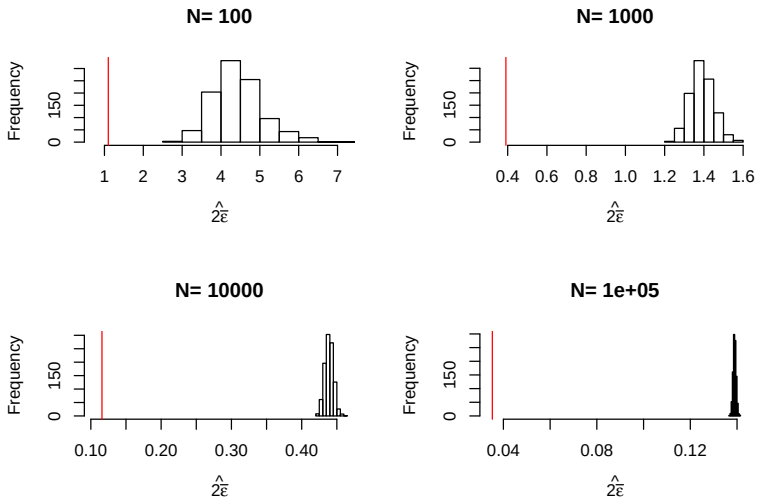


Figure 2.7: Distribution of the Chebyshev upper bound on sampling noise over replications of samples of different sizes (true sampling noise in red)

Figure 2.7 shows that the upper bound works: it is always bigger than the true sampling noise. Figure 2.7 also shows that the upper bound is large: it generally is of an order of magnitude bigger than the true sampling noise, and thus offers a blurry and too pessimistic view of the precision of an estimator. Figure 2.8 shows that the average Chebyshev bound gives an inflated estimate of sampling noise. Figure 2.9 shows that the Chebyshev confidence intervals are clearly less precise than the true unknown ones. With $N = 1000$, the true confidence intervals generally reject large negative effects, whereas the Chebyshev confidence intervals do not rule out this possibility. With $N = 10\hat{}\{4\}$, the true confidence intervals generally reject effects smaller than 0.1, whereas the Chebyshev confidence intervals cannot rule out small negative effects.

As a conclusion on Chebyshev estimates of sampling noise, their advantage is that they offer an upper bound on the noise: we can never underestimate noise if we use them. A downside of Chebyshev sampling noise estimates is their low precision, which makes it hard to pinpoint the true confidence intervals.

```
for (k in (1:length(N.sample))){
  table.noise$cheb.noise[k] <- mean(simuls.ww[[k]]$cheb.noise)
}
ggplot(table.noise, aes(x=as.factor(N), y=TT)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dod
  geom_errorbar(aes(ymin=TT-cheb.noise/2, ymax=TT+cheb.noise/2), width=.2,position=position_dodge(.9),c
  xlab("Sample Size")+
  theme_bw()
```

```
N.plot <- 40
plot.list <- list()

for (k in 1:length(N.sample)){
  set.seed(1234)
  test.cheb <- simuls.ww[[k]][sample(N.plot),c('WW','cheb.noise')]
  test.cheb <- as.data.frame(cbind(test.cheb,rep(samp.noise(simuls.ww[[k]][,'WW'],delta=delta),N.plot)))
```

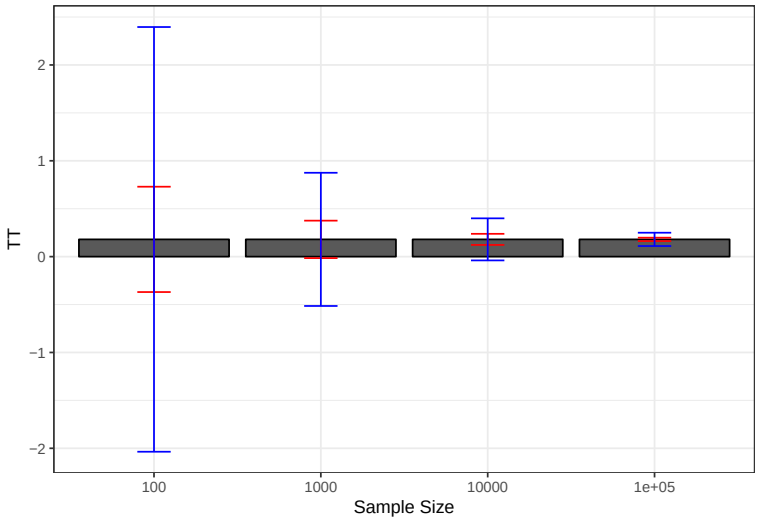


Figure 2.8: Average Chebyshev upper bound on sampling noise over replications of samples of different sizes (true sampling noise in red)

```
  colnames(test.cheb) <- c('WW','cheb.noise','sampling.noise')
  test.cheb$id <- 1:N.plot
  plot.test.cheb <- ggplot(test.cheb, aes(x=as.factor(id), y=WW)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
      geom_errorbar(aes(ymin=WW-cheb.noise/2, ymax=WW+cheb.noise/2), width=.2,position=position_dodge(.
      geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
      xlab("Sample id")+
      theme_bw()+
      ggtitle(paste("N=",N.sample[k]))
  plot.list[[k]] <- plot.test.cheb
}
plot.CI <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(N.sa
print(plot.CI)
```

2.2.3 Using the Central Limit Theorem

The main problem with Chebyshev's upper bound on sampling noise is that it is an upper bound, and thus it overestimates sampling noise and underestimates precision. One alternative to using Chebyshev's upper bound is to use the Central Limit Theorem (CLT). In econometrics and statistics, the CLT is used to derive approximate values for the sampling noise of estimators. Because these approximations become more and more precise as sample size increases, we call them asymptotic approximations.

Taken to its bare bones, the CLT states that the sum of i.i.d. random variables behaves approximately like a normal distribution when the sample size is large:

Theorem 2.4 (Central Limit Theorem)**.** *Let X_i be i.i.d. random variables with $\mathbb{E}[X_i] = \mu$ and $\mathbb{V}[X_i] = \sigma^2$, and define $Z_N = \frac{\frac{1}{N}\sum_{i=1}^{N} X_i - \mu}{\frac{\sigma}{\sqrt{N}}}$, then, for all z we have:*

$$\lim_{N\to\infty} \Pr(Z_N \leq z) = \Phi(z),$$

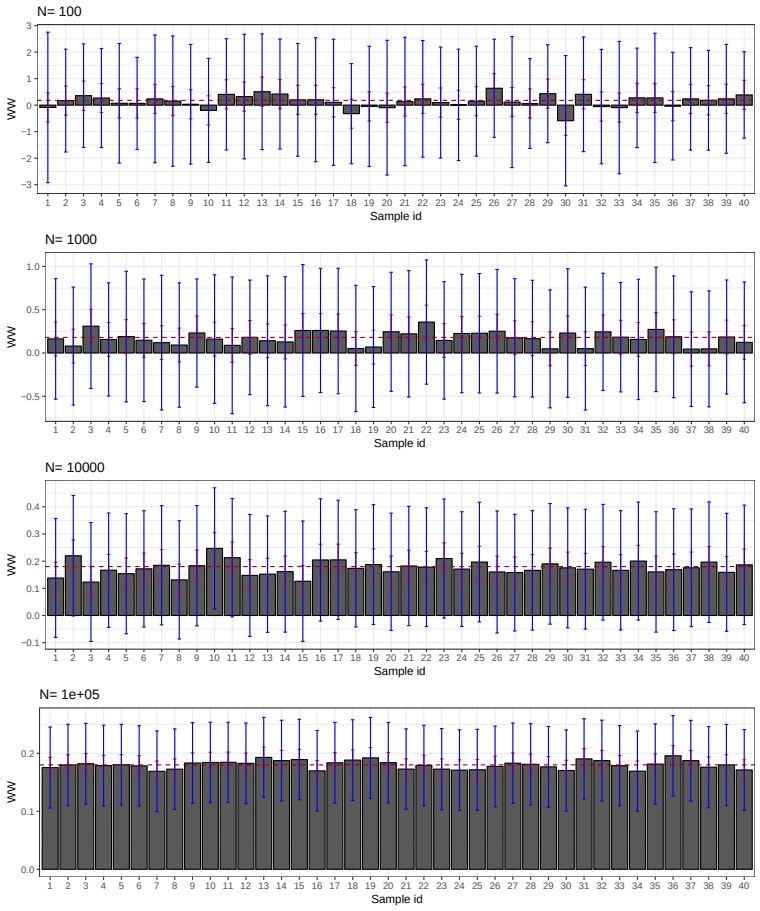


Figure 2.9: Chebyshev confidence intervals of $\hat{WW}$ for $\delta = 0.99$ over sample replications for various sample sizes (true confidence intervals in red)

where Φ is the cumulative distribution function of the centered standardized normal.

We say that Z_N converges in distribution to a standard normal random variable, and we denote: $Z_N \xrightarrow{d} \mathcal{N}(0,1)$.

The CLT is a beautiful result: the distribution of the average of realisations of any random variable that has finite mean and variance can be approximated by a normal when the sample size is large enough. The CLT is somehow limited though because not all estimators are sums. Estimators are generally more or less complex combinations of sums. In order to derive the asymptotic approximation for a lot of estimators that are combinations of sums, econometricians and statisticians complement the CLT with two other extremely powerful tools: Slutsky's theorem and the Delta method. Slutsky's theorem states that sums, products and ratios of sums that converge to a normal converge to the sum, product or ratio of these normals. The Delta method states that a function of a sum that converges to a normal converges to a normal whose variance is a quadratic form of the variance of the sum and of the first derivative of the function. Both of these tools are stated more rigorously in the appendix, but you do not need to know them for this class. The idea is for you to be aware of how the main approximations that we are going to use throughout this class have been derived.

Let me now state the main result of this section:
Theorem 2.5 (Asymptotic Estimate of Sampling Noise of WW)**.** *Under Assumptions 1.7, 2.3, 2.4 and 2.5, for a given confidence level δ and sample size N, the sampling noise of $\hat{WW}$ can be approximated as follows:*

$$2\epsilon \approx 2\Phi^{-1}\left(\frac{\delta+1}{2}\right)\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)}+\frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}} \equiv 2\tilde{\epsilon}.$$

Proof. See in Appendix A.1.2. □

Let's write an R function that computes this formula:

```
samp.noise.ww.CLT <- function(N,delta,v1,v0,p){
  return(2*qnorm((delta+1)/2)*sqrt((v1/p+v0/(1-p))/N))
}
```

Example 2.7. Let's see how the CLT performs in our example.

In our sample, for $\delta = 0.99$, the CLT estimate of sampling noise is $2\hat{\tilde{\epsilon}} = 0.35$. How does this compare with the true extent of sampling noise when $N = 1000$? Remember that we have computed an estimate of sampling noise out of our Monte Carlo replications. In Table 2.1, we can read that sampling noise is actually equal to 0.39. The CLT approximation is pretty precise: it only underestimates the true extent of sampling noise by 11%.

We can also compute the CLT approximation to sampling noise in all of our samples:

```
for (k in (1:length(N.sample))){
  simuls.ww[[k]]$CLT.noise <- samp.noise.ww.CLT(N.sample[[k]],delta,simuls.ww[[k]][,'V1'],simuls.ww[[k]]
}
par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww[[i]][,'CLT.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expression(hat(2*ba
  abline(v=table.noise[i,colnames(table.noise)=='sampling.noise'],col="red")
}
```

Figure 2.10 shows that the CLT works: CLT-based estimates of sampling noise approximates true sampling noise well. CLT-based approximations of sampling noise are even impressively accurate: they always capture the exact order of magnitude of sampling noise, although there is a slight underestimation when $N = 1000$ and 10^{4} and a slight overestimation when $N = 10^{5}$. This success should not come as a surprise as all shocks in our model are normally distributed, meaning that the CLT results are more than an approximation, they are exact. Results might be less spectacular when estimating the effect of the treatment on the outcomes in levels rather than in logs.

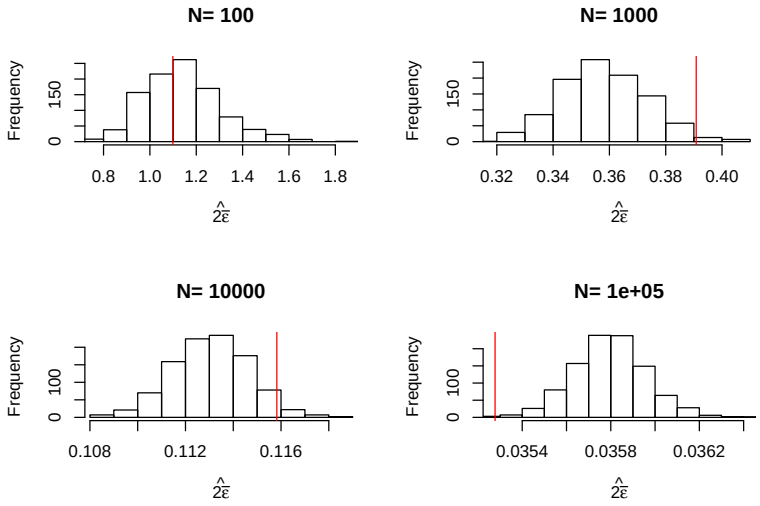


Figure 2.10: Distribution of the CLT approximation of sampling noise over replications of samples of different sizes (true sampling noise in red)

As a consequence, the average CLT-based estimates of sampling noise and of confidence intervals are pretty precise, as Figures 2.11 and 2.12 show. Let's pause for a second at the beauty of what we have achieved using the CLT: by using only information from one sample, we have been able to gauge extremely precisely how the estimator would behave over sampling repetitions.

```
for (k in (1:length(N.sample))){
  table.noise$CLT.noise[k] <- mean(simuls.ww[[k]]$CLT.noise)
}
ggplot(table.noise, aes(x=as.factor(N), y=TT)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dod
  geom_errorbar(aes(ymin=TT-CLT.noise/2, ymax=TT+CLT.noise/2), width=.2,position=position_dodge(.9),col
  xlab("Sample Size")+
  theme_bw()
```

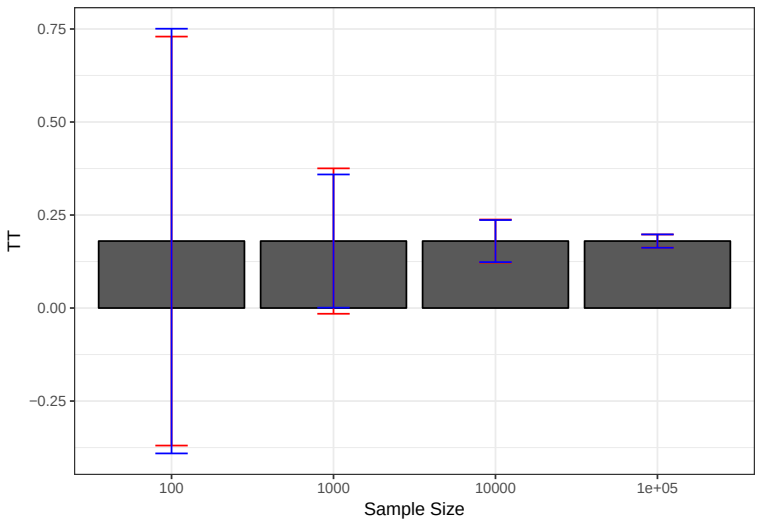


Figure 2.11: Average CLT-based approximations of sampling noise over replications of samples of different sizes (true sampling noise in red)

```
N.plot <- 40
plot.list <- list()

for (k in 1:length(N.sample)){
  set.seed(1234)
  test.CLT <- simuls.ww[[k]][sample(N.plot),c('WW','CLT.noise')]
  test.CLT <- as.data.frame(cbind(test.CLT,rep(samp.noise(simuls.ww[[k]][,'WW'],delta=delta),N.plot)))
  colnames(test.CLT) <- c('WW','CLT.noise','sampling.noise')
  test.CLT$id <- 1:N.plot
  plot.test.CLT <- ggplot(test.CLT, aes(x=as.factor(id), y=WW)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
      geom_errorbar(aes(ymin=WW-CLT.noise/2, ymax=WW+CLT.noise/2), width=.2,position=position_dodge(.9)
      geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
      xlab("Sample id")+
      theme_bw()+
      ggtitle(paste("N=",N.sample[k]))
  plot.list[[k]] <- plot.test.CLT
}
plot.CI <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(N.s
print(plot.CI)
```

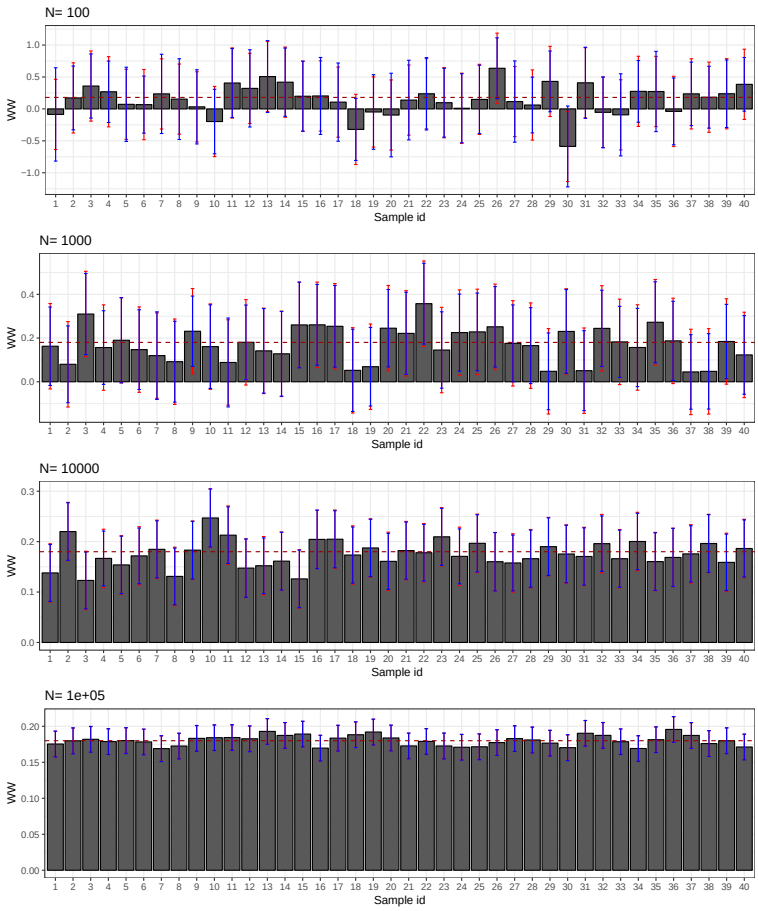


Figure 2.12: CLT-based confidence intervals of $\hat{WW}$ for $\delta = 0.99$ over sample replications for various sample sizes (true confidence intervals in red)

Remark. In proving the main result on the asymptotic distribution of $\hat{WW}$, we have also proved a very useful result: $\hat{WW}$ is the Ordinary Least Squares (OLS) estimator of β in the regression $Y_i = \alpha + \beta D_i + U_i$. This is pretty cool since we now can use our classical OLS estimator in our statistical package to estimate $\hat{WW}$. Let's compute the OLS estimate of WW in our sample:

```
ols.ww <- lm(y~Ds)
ww.ols <- ols.ww$coef[[2]]
```

We have $\hat{WW}_{OLS} = 0.13 = 0.13 = \hat{WW}$.
Remark. Another pretty cool consequence of Theorem 2.5 and of its proof is that the standard error of the OLS estimator of $\hat{WW}$ (σ_β) is related to the sampling noise of $\hat{WW}$ by the following formula: $2\tilde{\epsilon} = 2\Phi^{-1}\left(\frac{\delta+1}{2}\right)\sigma_\beta$.

This implies that sampling noise is equal to 5 σ_β when $\delta = 0.99$ and to 4 σ_β when $\delta = 0.95$. It is thus very easy to move from estimates of the standard error of the β coefficient to the extent of sampling noise.
Remark. A last important consequence of Theorem 2.5 and of its proof is that the standard error of the OLS estimator of $\hat{WW}$ (σ_β) that we use is the heteroskedasticity-robust one.

Using the RCM, we can indeed show that:

$$\begin{aligned}
\alpha &= \mathbb{E}[Y_i^0|D_i = 0] \\
\beta &= \Delta_{TT}^Y \\
U_i &= Y_i^0 - \mathbb{E}[Y_i^0|D_i = 0] + D_i(\Delta_i^Y - \Delta_{TT}^Y),
\end{aligned}$$

Under Assumption 1.7, we have:

$$U_i = (1 - D_i)(Y_i^0 - \mathbb{E}[Y_i^0|D_i = 0]) + D_i(Y_i^1 - \mathbb{E}[Y_i^1|D_i = 1])$$

There is heteroskedasticity because the outcomes of the treated and of the untreated have different variances:

$$\begin{aligned}
\mathbb{V}[U_i|D_i = d] &= \mathbb{E}[U_i^2|D_i = d] \\
&= \mathbb{E}[(Y_i^d - \mathbb{E}[Y_i^d|D_i = d])^2|D_i = d] \\
&= \mathbb{V}[Y_i^d|D_i = d]
\end{aligned}$$

We do not want to assume homoskedasticity, since it would imply a constant treatment effect. Indeed, $\mathbb{V}[Y_i^1|D_i = 1] = \mathbb{V}[Y_i^0|D_i = 1] + \mathbb{V}[\alpha_i|D_i = 1]$.
Remark. In order to estimate the heteroskedasticity robust standard error from the OLS regression, we can use the sandwich package in R Most available heteroskedasticity robust estimators based on the CLT can be written in the following way:

$$\mathbb{V}[\hat{\Theta}_{OLS}] \approx (X'X)^{-1}X'\hat{\Omega}X(X'X)^{-1},$$

where X is the matrix of regressors and $\hat{\Omega} = \text{diag}(\hat{\sigma}_{U_1}^2, \ldots, \hat{\sigma}_{U_N}^2)$ is an estimate the covariance matrix of the residuals U_i. Here are various classical estimators for $\hat{\Omega}$:

$$\begin{aligned}
&\text{HC0:} && \hat{\sigma_{U_i}}^2 = \hat{U_i}^2 \\
&\text{HC1:} && \hat{\sigma_{U_i}}^2 = \frac{N}{N-K}\hat{U_i}^2 \\
&\text{HC2:} && \hat{\sigma_{U_i}}^2 = \frac{\hat{U_i}^2}{1-h_i} \\
&\text{HC3:} && \hat{\sigma_{U_i}}^2 = \frac{\hat{U_i}^2}{(1-h_i)^2},
\end{aligned}$$

where $\hat{U}_i$ is the residual from the OLS regression, K is the number of regressors, h_i is the leverage of observation i, and is the i^{th} diagonal element of $H = X(X'X)^{-1}X'$. HC1 is the one reported by Stata when using the 'robust' option.

Example 2.8. Using the sandwich package, we can estimate the heteroskedasticity-robust variance-covariance matrix and sampling noise as follows:

```
ols.ww.vcov.HC0 <- vcovHC(ols.ww, type = "HC0")
samp.noise.ww.CLT.ols <- function(delta,reg,...){
  return(2*qnorm((delta+1)/2)*sqrt(vcovHC(reg,...)[2,2]))
}
```

For $\delta = 0.99$, sampling noise estimated using the "HC0" option is equal to 0.35. This is exactly the value we have estimated using our CLT-based formula ($2\widehat{\tilde{\epsilon}} = 0.35$). Remember that sampling noise is actually equal to 0.39. Other "HC" options might be better in small samples. For example, with the "HC1" option, we have an estimate for sampling noise of 0.35. What would have happened to our estimate of sampling noise if we had ignored heteroskedasticity? The default OLS standard error estimate yields an estimate for sampling noise of 0.36.

2.2.4 Using resampling methods

The main intuition behind resampling methods is to use the sample as a population, to draw samples from it and compute our estimator on each of these samples in order to gauge its variability over sampling repetitions. There are three main methods of resampling that work that way: bootstrapping, radomization inference and subsampling. Bootstrapping draws samples with replacement, so that each sample has the same size as the original sample. Subsampling draws samples without replacement, thereby the samples are of a smaller size than the original one. Randomization inference keeps the same sample in all repetitions, but changes the allocation of the treatment.

Why would we use resampling methods instead of CLT-based standard errors? There are several possible reasons:

1. Asymptotic refinements: sometimes, resampling methods are more precise in small samples than the CLT-based asymptotic approaches. In that case, we say that resampling methods offer asymptotic refinements.
2. Ease of computation: for some estimators, the CLT-based estimates of sampling noise are complex or cumbersome to compute, whereas resampling methods are only computationally intensive.
3. Inexistence of CLT-based estimates of sampling noise: some estimators do not have any CLT-based estimates of sampling noise yet. That was the case for the Nearest-Neighbour Matching estimator (NNM) for a long time for example. It still is the case for the Synthetic Control Method estimator. Beware though that the bootstrap is not valid for all estimators. For example, it is possible to show that the bootstrap is invalid for NNM. Subsampling is valid for NNM though (see Abadie and Imbens, 2006).

2.2.4.1 Bootstrap

The basic idea of the bootstrap is to use Monte Carlo replications to draw samples from the original sample with replacement. Then, at each replication, we compute the value of our estimator $\hat{E}$ on the new sample. Let's call this new value $\hat{E}_k^*$ for bootstrap replication k. Under certain conditions, the distribution of $\hat{E}_k^*$ approximates the distribution of $\hat{E}$ over sample repetitions very well, and all the more so as the sample size gets large.

What are the conditions under which the bootstrap is going to provide an accurate estimation of the distribution of $\hat{E}$? Horowitz (2001) reports on a very nice result by Mammen that makes these conditions clear:

Theorem 2.6 (Mammen (1992))**.** *Let $\{X_i : i = 1, \dots, N\}$ be a random sample from a population. For a sequence of functions g_N and sequences of numbers t_N and σ_N, define $\bar{g}_N = \frac{1}{N}\sum_{i=1}^N g_N(X_i)$ and $T_N = (\bar{g}_N - t_N)/\sigma_N$. For the bootstrap sample $\{X_i^* : i = 1, \dots, N\}$, define $\bar{g}_N^* = \frac{1}{N}\sum_{i=1}^N g_N(X_i^*)$ and $T_N^* = (\bar{g}_N^* - \bar{g}_N)/\sigma_N$. Let $G_N(\tau) = \Pr(T_N \leq \tau)$ and $G_N^*(\tau) = \Pr(T_N^* \leq \tau)$, where this last probability distribution is taken over bootstrap sampling replications. Then G_N^* consistently estimates G_N if and only if $T_N \xrightarrow{d} \mathcal{N}(0,1)$.*

Theorem 2.6 states that the bootstrap will offer a consistent estimation of the distribution of a given estimator if and only if this estimator is asymptotically normally distributed. It means that we could theoretically use the CLT-based asymptotic distribution to compute sampling noise. So, and it demands to be stronlgy emphasized, **the bootstrap is not valid when the CLT fails**.

How do we estimate sampling noise with the bootstrap? There are several ways to do so, but I am going to emphasize the most widespread here, that is known as the percentile method. Let's define $E^*_{\frac{1-\delta}{2}}$ and $E^*_{\frac{1+\delta}{2}}$ as the corresponding quantiles of the bootstrap distribution of $\hat{E}_k^*$ over a large number K of replications. The bootstrapped sampling noise using the percentile method is simply the distance between these two quantities.

Theorem 2.7 (Bootstrapped Estimate of Sampling Noise of WW)**.** *Under Assumptions 1.7, 2.3, 2.4 and 2.5, for a given confidence level δ and sample size N, the sampling noise of $\hat{WW}$ can be approximated as follows:*

$$2\epsilon \approx E^*_{\frac{1+\delta}{2}} - E^*_{\frac{1-\delta}{2}} \equiv 2\bar{\epsilon}^b.$$

Proof. The WW estimator can be written as a sum:

$$\Delta^{\hat{Y}}_{WW} = \frac{1}{N}\sum_{i=1}^N \frac{\left(Y_i - \frac{1}{N}\sum_{i=1}^N Y_i\right)\left(D_i - \frac{1}{N}\sum_{i=1}^N D_i\right)}{\frac{1}{N}\sum_{i=1}^N \left(D_i - \frac{1}{N}\sum_{i=1}^N D_i\right)^2}.$$

Using Lemma A.5, we know that the WW estimator is asymptotically normal under Assumptions 1.7, 2.3, 2.4 and 2.5. Using Theorem 2.6 proves the result. □

Remark. With the bootstrap, we are not going to define the confidence interval using Theorem 2.1 but directly using $\left\{E^*_{\frac{1-\delta}{2}}; E^*_{\frac{1+\delta}{2}}\right\}$. Indeed, we have defined the bootstrapped estimator of sampling noise by using the asymetric confidence interval. We could have used the equivalent of Definition 2.1 on the bootstrapped samples to compute sampling noise using the symmetric confidence interval. Both are feasible and similar in large samples, since the asymptotic distribution is symmetric. One advantage of asymetric confidence intervals is that they might capture deviations from the normal distribution in small samples. These advantages are part of what we call asymptotic refinements. Rigorously, though, asymptotic refinements have not been proved to exist for the percentile method but only for the method bootstrapping asymptotically pivotal quantities.

Remark. We say that a method brings asymptotic refinements if it increases the precision when estimating sampling noise and confidence intervals relative to the asymptotic CLT-based approximation. The bootstrap has been shown rigorously to bring asymptotic refinements when used to estimate the distribution of asymptotically pivotal statistic. An asymptotically pivotal statistic is a statistic that can be computed from the sample but that, asymptotically, converges to a quantity that does not depend on the sample, like for example a standard normal. Using Lemma A.5, we know for example that the following statistic is asymptotically normal:

$$T_N^{WW} = \frac{\Delta_{WW}^{\hat{Y}} - \Delta_{TT}^{Y}}{\sqrt{\frac{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}{N}}} \xrightarrow{d} \mathcal{N}(0,1).$$

To build a confidence interval bootstrapping T_N^{WW}, compute an estimator of T_N^{WW} for each bootstrapped sample, say $\hat{T}_{N,k}^{WW*}$. You can for example use the OLS estimator in the bootstrapped sample, with a heteroskedasticity-robust standard error estimator. Or you can compute the WW estimator by hand in the sample along with an estimator of its variance using the variance of the outcomes in the treated and control groups. You can then estimate the confidence interval as follows: $\left\{\Delta_{WW}^{\hat{Y}} - \sigma_{\hat{WW}}\hat{T}_{N,\frac{1-\delta}{2}}^{WW*}; \Delta_{WW}^{\hat{Y}} + \sigma_{\hat{WW}}\hat{T}_{N,\frac{1+\delta}{2}}^{WW*}\right\}$, where $\hat{T}_{N,q}^{WW*}$ iq the q^{th} quantile of the distribution of $\hat{T}_{N,k}^{WW*}$ over sampling replications and $\sigma_{\hat{WW}}$ is an estimate of the variance of $\Delta_{WW}^{\hat{Y}}$ (either the CLT-based approximation of the bootstrapped one, see below).

Remark. One last possibility to develop an estimator for sampling noise and confidence interval is to use the bootstrap in order to estimate the variance of the estimator $\hat{E}$, $\hat{\sigma_E^2}$, and then use it to compute sampling noise. If $\hat{E}$ is asymptotically normally distributed, we have that sampling noise is equal to $2\Phi^{-1}\left(\frac{\delta+1}{2}\right)\hat{\sigma_E}$. You can use the usual formula from Theore 2.1 to compute the confidence interval. The bootstrapped variance of $\hat{E}$, $\hat{\sigma_E^2}$, is simply the variance of $\hat{E}_k^*$ over bootstrap replications.

Example 2.9. In the numerical example, I am going to derive the bootstrapped confidence intervals and sampling noise for the percentile method. Let's first put the dataset from our example in a nice data frame format so that resampling is made easier. We then define a function taking a number of bootstrapped replications and spitting out sampling noise and confidence intervals.

```
data <- as.data.frame(cbind(y,Ds,yB))
boot.fun.ww.1 <- function(seed,data){
  set.seed(seed,kind="Wichmann-Hill")
  data <- data[sample(nrow(data),replace = TRUE),]
  ols.ww <- lm(y~Ds,data=data)
  ww <- ols.ww$coef[[2]]
  return(ww)
}

boot.fun.ww <- function(Nboot,data){
  #sfInit(parallel=TRUE,cpus=8)
  boot <- lapply(1:Nboot,boot.fun.ww.1,data=data)
  #sfStop()
  return(unlist(boot))
}

boot.CI.ww <- function(boot,delta){
  return(c(quantile(boot,prob=(1-delta)/2),quantile(boot,prob=(1+delta)/2)))
}

boot.samp.noise.ww <- function(boot,delta){
  return(quantile(boot,prob=(1+delta)/2)-quantile(boot,prob=(1-delta)/2))
```

```
}

Nboot <- 500
ww.boot <- boot.fun.ww(Nboot,data)
ww.CI.boot <- boot.CI.ww(ww.boot,delta)
ww.samp.noise.boot <- boot.samp.noise.ww(ww.boot,delta)
```

Over 500 replications, the 99% bootstrapped confidence interval using the percentile method is $\{-0.023; 0.316\}$. As a consequence, the bootstrapped estimate of 99% sampling noise is of 0.339. Remember that, with $N = 1000$, sampling noise is actually equal to 0.39.

In order to assess the global precision of bootstrapping, we are going to resort to Monte Carlo simulations. For each Monte Carlo sample, we are going to estimate sampling noise and confidence intervals using the bootstrap. As you can imagine, this is going to prove rather computationally intensive. I cannot use parallelization twice: I have to choose whether to parallelize the Monte Carlo simulations or the bootstrap simulations. I have choosen to parallelize the outer loop, so that a given job takes longer on each cluster.

```
monte.carlo.ww.boot <- function(s,N,param,Nboot,delta){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  Ds <- rep(0,N)
  V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
  Ds[V<=log(param["barY"])] <- 1
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  data <- as.data.frame(cbind(y,Ds,yB))
  ww.boot <- boot.fun.ww(Nboot,data)
  ww.CI.boot <- boot.CI.ww(ww.boot,delta)
  ww.samp.noise.boot <- boot.samp.noise.ww(ww.boot,delta)
  return(c((1/sum(Ds))*sum(y*Ds)-(1/sum(1-Ds))*sum(y*(1-Ds)),var(y[Ds==1]),var(y[Ds==0]),mean(Ds),ww.CI
}

sf.simuls.ww.N.boot <- function(N,Nsim,Nboot,delta,param){
  sfInit(parallel=TRUE,cpus=2*ncpus)
  sfExport("boot.fun.ww","boot.CI.ww","boot.samp.noise.ww","boot.fun.ww.1")
  sim <- as.data.frame(matrix(unlist(sfLapply(1:Nsim,monte.carlo.ww.boot,N=N,Nboot=Nboot,delta=delta,pa
  sfStop()
  colnames(sim) <- c('WW','V1','V0','p','boot.lCI','boot.uCI','boot.samp.noise')
  return(sim)
}

simuls.ww.boot <- lapply(N.sample,sf.simuls.ww.N.boot,Nsim=Nsim,param=param,Nboot=Nboot,delta=delta)
```

We can now graph our bootstrapped estimate of sampling noise in all of our samples, the average bootstrapped

estimates of sampling noise and of confidence intervals, in Figures 2.13, 2.14 and 2.15 show.

```
par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww.boot[[i]][,'boot.samp.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expressi
  abline(v=table.noise[i,colnames(table.noise)=='sampling.noise'],col="red")
}
```

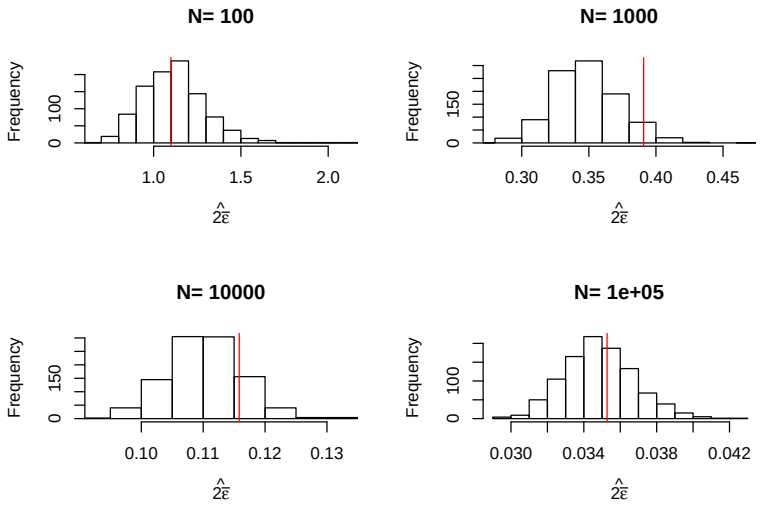


Figure 2.13: Distribution of the bootstrapped approximation of sampling noise over replications of samples of different sizes (true sampling noise in red)

```
for (k in (1:length(N.sample))){
  table.noise$boot.noise[k] <- mean(simuls.ww.boot[[k]]$boot.samp.noise)
}
ggplot(table.noise, aes(x=as.factor(N), y=TT)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dod
  geom_errorbar(aes(ymin=TT-boot.noise/2, ymax=TT+boot.noise/2), width=.2,position=position_dodge(.9),c
  xlab("Sample Size")+
  theme_bw()
```

```
N.plot <- 40
plot.list <- list()

for (k in 1:length(N.sample)){
 set.seed(1234)
 test.boot <- simuls.ww.boot[[k]][sample(N.plot),c('WW','boot.lCI','boot.uCI')]
 test.boot <- as.data.frame(cbind(test.boot,rep(samp.noise(simuls.ww.boot[[k]][,'WW'],delta=delta),N.pl
 colnames(test.boot) <- c('WW','boot.lCI','boot.uCI','sampling.noise')
 test.boot$id <- 1:N.plot
 plot.test.boot <- ggplot(test.boot, aes(x=as.factor(id), y=WW)) +
     geom_bar(position=position_dodge(), stat="identity", colour='black') +
     geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
     geom_errorbar(aes(ymin=boot.lCI, ymax=boot.uCI), width=.2,position=position_dodge(.9),color='blue'
     geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
     xlab("Sample id")+
     theme_bw()+
     ggtitle(paste("N=",N.sample[k]))
 plot.list[[k]] <- plot.test.boot
```

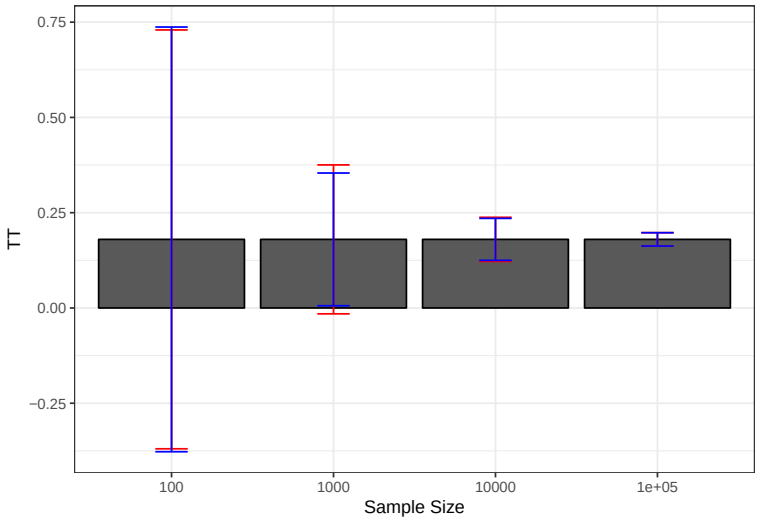


Figure 2.14: Average bootstrapped approximations of sampling noise over replications of samples of different sizes (true sampling noise in red)

```
}
plot.CI <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(N.s
print(plot.CI)
```

TO DO: COMMENT AND USE PIVOTAL TEST STATISTIC

2.2.4.2 Randomization inference

Randomization inference (a.k.a. Fisher's permutation approach) tries to mimick the sampling noise due to the random allocation of the treatment vector, as we have seen in Section 2.1.3. In practice, the idea is simply to look at how the treatment effect that we estimate varies when we visit all the possible allocations of the treament dummy in the sample. For each new allocation, we are going to compute the with/without estimator using the observed outcomes and the newly allocated treatment dummy. It means that some actually treated observations are going to enter into the computation of the control group mean, while some actually untreated observations are going to enter into the computation of the treatment group mean. As a consequence, the resulting distribution will be centered at zero. Under the assumption of a constant treatment effect, the distribution of the parameter obtained using randomization inference will be an exact estimation of sampling noise for the sample treatment effect.

Notice how beautilful the result is: randomization inference yields an **exact** measure of sampling noise. The resulting estimate of sampling noise is not an approximation that is going to become better as sample size increases. No, it is the **actual** value of sampling noise in the sample.

There are two ways to compute a confidence interval using Fisher's permutation approach. One is to form symmetric intervals using our estimate of sampling noise as presented in Section 2.1.4. Another approach is to directly use the quantiles of the distribution of the parameter centered around the estimated treatment effect, in the same spirit as bootstrapped confidence intervals using the percentile approach. This last approach accomodates possible asymetries in the finite sample distribution of the treatment effect.

Computing the value of the treatment effect for all possible treatment allocations can take a lot of time with large samples. That's why we in general compute the test statistic for a reasonably large number of random allocations.

Remark. Fisher's original approach is slightly different from the one I delineate here. Fisher wanted to derive a test statistic for whether the treatment effect was zero, not to estimate sampling noise. Under the null that

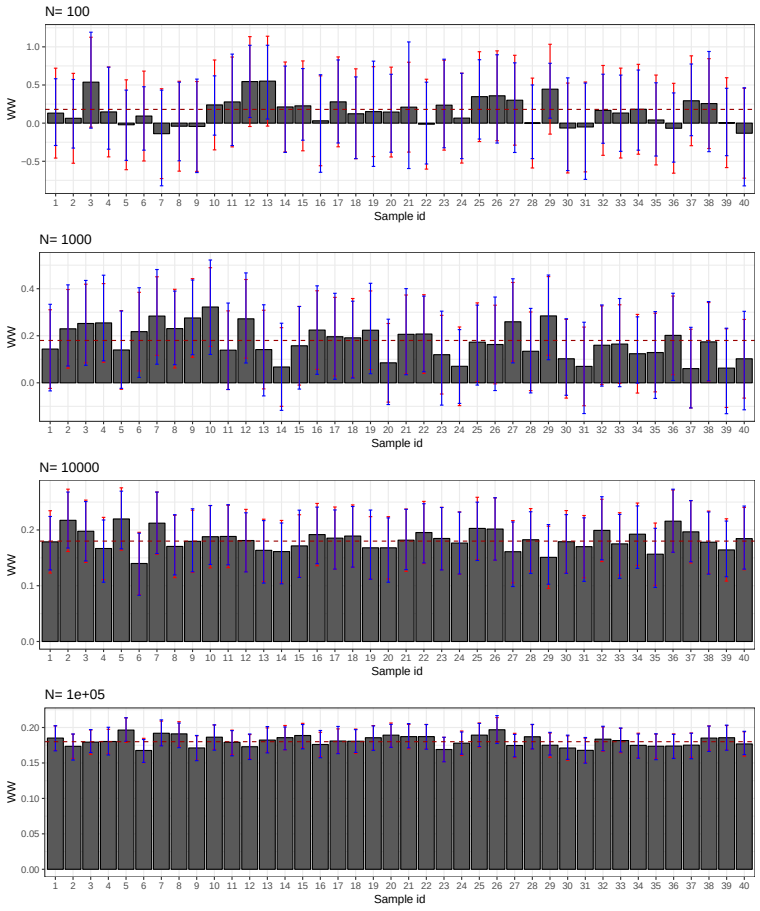


Figure 2.15: Bootstrapped confidence intervals of $\hat{WW}$ for $\delta = 0.99$ over sample replications for various sample sizes (true confidence intervals in red)

the treatment has absolutely no effect whatsoever on any unit, any test statistic whose value should be zero if the two distributions where identical can be computed on the actual sample and its distribution can be derived using Fisher's permutation approach. The test statistic can be the difference in means, standard deviations, medians, ranks, the T-stat, the Kolmogorov-Smirnov test statistic or any other test statistic that you might want to compute. Comparing the actual value of the test statistic to its distribution under the null gives a p-value for the validity of the null.

Remark. Imbens and Rubin propose a more complex procedure to derive the confidence interval for the treatment effect using randomization inference. They propose to compute Fisher's p-value for different values of the treatment effect, and to set the confidence interval as the values of the treatment effect under and above which the p-value is smaller than δ. When using the with/without estimator as the test statistic, the two approches should be equivalent. Is is possible that the estimates using statistics less influenced by outliers are more precise though.

Remark. Note that we pay two prices for having an exact estimation of sampling noise:

1. We have to assume that the treatment effect is constant, e.g. we have to assume homoskedasticity. This is in general not the case. Whether this is in general a big issue depends on how large the difference is between homoskedastic and heteroskedastic standard errors. One way around this issue would be to add a small amount of noise to the observations that are in the group with the lowest variance. Whether this would work in practice is still to be demonstrated.
2. We have to be interested only in the sampling noise of the sample treatment effect. The sampling noise of the population treatment effect is not estimated using Fisher's permutation approach. As we have seen in Section 2.1.3, there is no practical difference between these two sampling noises in our example. Whether this is the case in general deserves further investigation.

Example 2.10. In practice, randomization inference is very close to a bootstrap procedure, except that instead of resampling with replacement from the original sample, we only change the vector of treatment allocation at each replication.

```
fisher.fun.ww.1 <- function(seed,data){
  set.seed(seed,kind="Wichmann-Hill")
  data$D <- rbinom(nrow(data),1,mean(data$Ds))
  ols.ww <- lm(y~D,data=data)
  ww <- ols.ww$coef[[2]]
  return(ww)
}

fisher.fun.ww <- function(Nfisher,data,delta){
  fisher <- unlist(lapply(1:Nfisher,fisher.fun.ww.1,data=data))
  ols.ww <- lm(y~Ds,data=data)
  ww <- ols.ww$coef[[2]]
  fisher <- fisher+ ww
  fisher.CI.ww <- c(quantile(fisher,prob=(1-delta)/2),quantile(fisher,prob=(1+delta)/2))
  fisher.samp.noise.ww <- quantile(fisher,prob=(1+delta)/2)-quantile(fisher,prob=(1-delta)/2)
  return(list(fisher,fisher.CI.ww,fisher.samp.noise.ww))
}

Nfisher <- 500
ww.fisher <- fisher.fun.ww(Nfisher,data,delta)
```

Over 500 replications, the 99% confidence interval based on Fisher's permutation approach is $\{-0.052; 0.3\}$. As a consequence, the bootstrapped estimate of 99% sampling noise is of 0.352. Remember that, with $N = 1000$, sampling noise is actually equal to 0.39.

In order to assess the global precision of Fisher's permutation method, we are going to resort to Monte Carlo simulations.

```
monte.carlo.ww.fisher <- function(s,N,param,Nfisher,delta){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  Ds <- rep(0,N)
  V <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]+param["sigma2U"]))
  Ds[V<=log(param["barY"])] <- 1
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  data <- as.data.frame(cbind(y,Ds,yB))
  ww.fisher <- fisher.fun.ww(Nfisher,data,delta)
  return(c((1/sum(Ds))*sum(y*Ds)-(1/sum(1-Ds))*sum(y*(1-Ds)),var(y[Ds==1]),var(y[Ds==0]),mean(Ds),ww.fis
}

sf.simuls.ww.N.fisher <- function(N,Nsim,Nfisher,delta,param){
  sfInit(parallel=TRUE,cpus=2*ncpus)
  sfExport("fisher.fun.ww","fisher.fun.ww.1")
  sim <- as.data.frame(matrix(unlist(sfLapply(1:Nsim,monte.carlo.ww.fisher,N=N,Nfisher=Nfisher,delta=del
  sfStop()
  colnames(sim) <- c('WW','V1','V0','p','fisher.lCI','fisher.uCI','fisher.samp.noise')
  return(sim)
}

simuls.ww.fisher <- lapply(N.sample,sf.simuls.ww.N.fisher,Nsim=Nsim,param=param,Nfisher=Nfisher,delta=de
```

We can now graph our bootstrapped estimate of sampling noise in all of our samples, the average bootstrapped estimates of sampling noise and of confidence intervals, as Figures 2.16, 2.17 and 2.18 show. The results are pretty good. On average, estimates of sampling noise using Randomization Inference are pretty accurate, as Figure 2.17 shows. It seems that sampling noise is underestimated by Randomization Inference when $N = 1000$, without any clear reason why.

```
par(mfrow=c(2,2))
for (i in 1:4){
  hist(simuls.ww.fisher[[i]][,'fisher.samp.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expre
  abline(v=table.noise[i,colnames(table.noise)=='sampling.noise'],col="red")
}
```

```
for (k in (1:length(N.sample))){
  table.noise$fisher.noise[k] <- mean(simuls.ww.fisher[[k]]$fisher.samp.noise)
}
ggplot(table.noise, aes(x=as.factor(N), y=TT)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
  geom_errorbar(aes(ymin=TT-sampling.noise/2, ymax=TT+sampling.noise/2), width=.2,position=position_dodg
  geom_errorbar(aes(ymin=TT-fisher.noise/2, ymax=TT+fisher.noise/2), width=.2,position=position_dodge(.
```

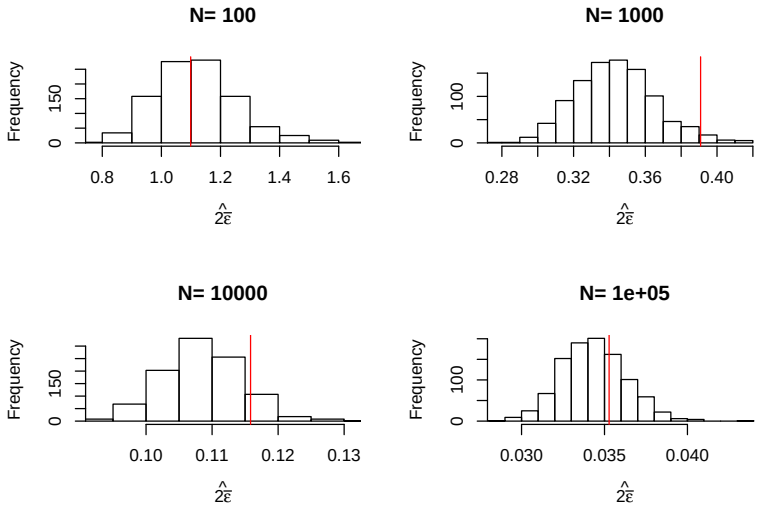


Figure 2.16: Distribution of the estimates of sampling noise using Randomization Inference over replications of samples of different sizes (true sampling noise in red)

```
  xlab("Sample Size")+
  theme_bw()
```

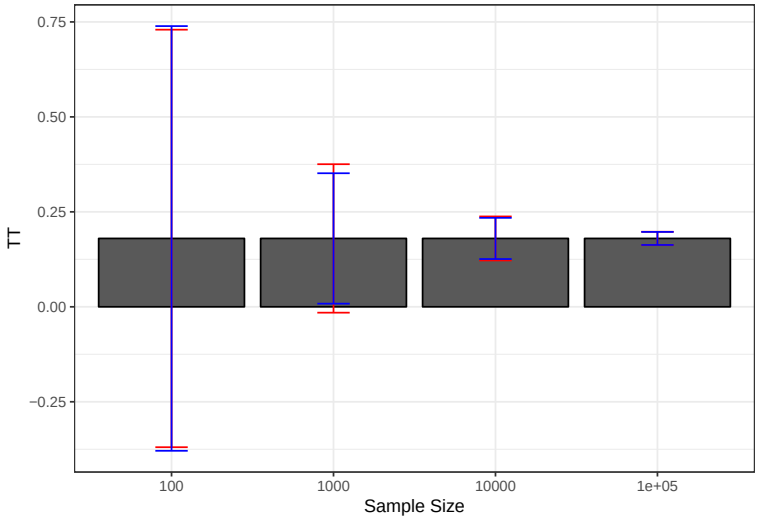


Figure 2.17: Average estimates of sampling noise using Randomization Inference over replications of samples of different sizes (true sampling noise in red)

```
N.plot <- 40
plot.list <- list()

for (k in 1:length(N.sample)){
 set.seed(1234)
 test.fisher <- simuls.ww.fisher[[k]][sample(N.plot),c('WW','fisher.lCI','fisher.uCI')]
 test.fisher <- as.data.frame(cbind(test.fisher,rep(samp.noise(simuls.ww.fisher[[k]][,'WW'],delta=delta
 colnames(test.fisher) <- c('WW','fisher.lCI','fisher.uCI','sampling.noise')
 test.fisher$id <- 1:N.plot
 plot.test.fisher <- ggplot(test.fisher, aes(x=as.factor(id), y=WW)) +
```

```
    geom_bar(position=position_dodge(), stat="identity", colour='black') +
    geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
    geom_errorbar(aes(ymin=fisher.lCI, ymax=fisher.uCI), width=.2,position=position_dodge(.9),color='bl
    geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
    xlab("Sample id")+
    theme_bw()+
    ggtitle(paste("N=",N.sample[k]))
 plot.list[[k]] <- plot.test.fisher
}
plot.CI <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(N.sa
print(plot.CI)
```

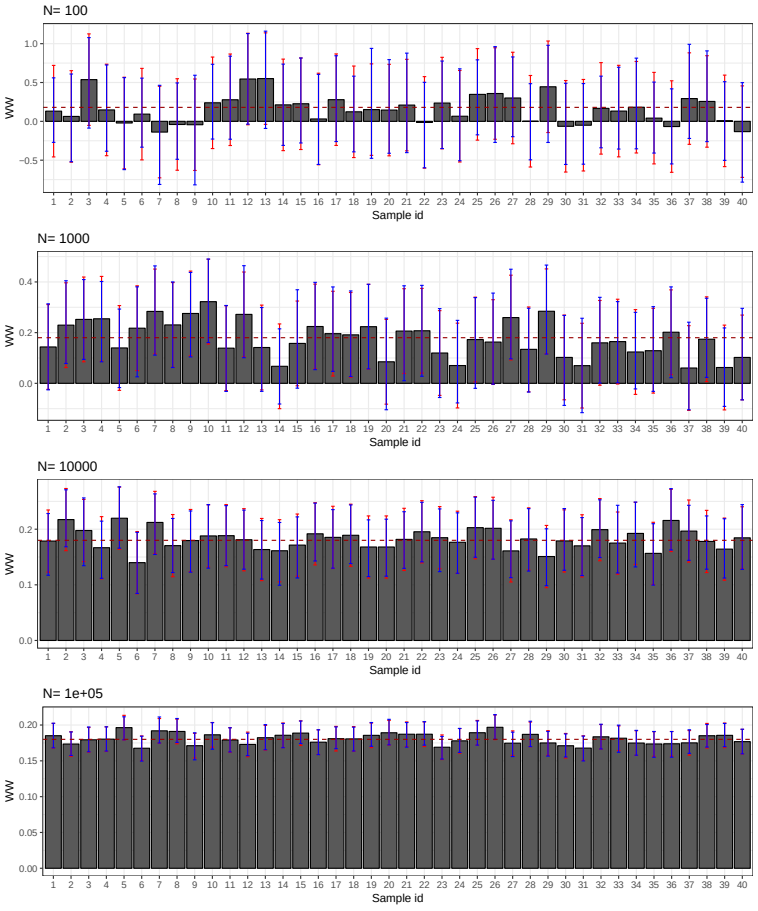


Figure 2.18: Confidence intervals of $\hat{WW}$ for $\delta = 0.99$ estimated using Randomization Inference over sample replications for various sample sizes (true confidence intervals in red)

TO DO: ALTERNATIVE APPROACH USING p-VALUES

2.2.4.3 Subsampling

TO DO: ALL

Part II

Methods of Causal Inference

Chapter 3

Randomized Controlled Trials

The most robust and rigorous method that has been devised by social scientists to estimate the effect of an intervention on an outcome is the Randomized Controlled Trial (RCT). RCTs are used extensively in the field to evaluate a wide array of programs, from development, labor and education interventions to environmental nudges to website and search engine features.

The key feature of an RCT is the introduction by the researcher of randomness in the allocation of the treatment. Individuals with $R_i = 1$, where R_i denotes the outcome of a random event, such as a coin toss, have a higher probability of receiving the treatment. Potential outcomes have the same distribution in both $R_i = 1$ and $R_i = 0$ groups. If we observe different outcomes between the treatment and control group, it has to be because of the causal effect of the treatment, since both groups only differ by the proportion of treated and controls.

The most attractive feature of RCTs is that researchers enforce the main identification assumption (we do not have to assume that it holds, we can make sure that it does). This property of RCTs distinguishes them from all the other methods that we are going to learn in this class.

In this lecture, we are going to study how to estimate the effect of an intervention on an outcome using RCTs. We are especially going to study the various types of designs and what can be recovered from them using which technique. For each design, we are going to detail which treatment effect it enables us to identify, how to obtain a sample estimate of this treatment effect and how to estimate the associated sampling noise. The main substantial difference between these four designs are the types of treatment effect parameters that they enable us to recover. Sections 3.1 to 3.4 of this lecture introduces the four designs and how to analyze them.

Unfortunately, RCTs are not bullet proof. They suffer from problems that might make their estimates of causal effects badly biased. Section 3.5 surveys the various threats and what we can do to try to minimize them.

3.1 Brute Force Design

In the Brute Force Design, eligible individuals are randomly assigned to the treatment irrespective of their willingness to accept it and have to comply with the assignment. This is a rather dumb procedure but it is very easy to analyze and that is why I start with it. With the Brute Force Design, you can recover the average effect of the treatment on the whole population. This parameter is generally called the Average Treatment Effect (ATE).

In this section, I am going to detail the assumptions required for the Brute Force Design to identify the ATE, how to form an estimator of the ATE and how to estimate its sampling noise.

3.1.1 Identification

In the Brute Force Design, we need two assumptions for the ATE to be identified in the population: Independence and Brute Force Validity.

Definition 3.1 (Independence)**.** We assume that the allocation of the program is independent of potential outcomes:

$$R_i \perp\!\!\!\perp (Y_i^0, Y_i^1).$$

Here, $\perp\!\!\!\perp$ codes for independence or random variables. Independence can be enforced by the randomized allocation.

We need a second assumption for the Brute Force Design to work:

Definition 3.2 (Brute Force Validity)**.** We assume that the randomized allocation of the program is mandatory and does not interfere with how potential outcomes are generated:

$$Y_i = \begin{cases} Y_i^1 & \text{if } R_i = 1 \\ Y_i^0 & \text{if } R_i = 0 \end{cases}$$

with Y_i^1 and Y_i^0 the same potential outcomes as defined in Lecture~0 with a routine allocation of the treatment.

Under both Idependence and Brute Force Validity, we have the follwing result:

Theorem 3.1 (Identification in the Brute Force Design)**.** *Under Assumptions 3.1 and 3.2, the WW estimator identifies the Average Effect of the Treatment (ATE):*

$$\Delta_{WW}^Y = \Delta_{ATE}^Y,$$

with:

$$\begin{aligned} \Delta_{WW}^Y &= \mathbb{E}[Y_i|R_i = 1] - \mathbb{E}[Y_i|R_i = 0] \\ \Delta_{ATE}^Y &= \mathbb{E}[Y_i^1 - Y_i^0]. \end{aligned}$$

Proof.

$$\begin{aligned} \Delta_{WW}^Y &= \mathbb{E}[Y_i|R_i = 1] - \mathbb{E}[Y_i|R_i = 0] \\ &= \mathbb{E}[Y_i^1|R_i = 1] - \mathbb{E}[Y_i^0|R_i = 0] \\ &= \mathbb{E}[Y_i^1] - \mathbb{E}[Y_i^0] \\ &= \mathbb{E}[Y_i^1 - Y_i^0], \end{aligned}$$

where the first equality uses Assumption 3.2, the second equality Assumption 3.1 and the last equality the linearity of the expectation operator. □

Remark. As you can see from Theorem 3.1, ATE is the average effect of the treatment on the whole population, those who would be eligible for it and those who would not. ATE differs from TT because the effect of the treatment might be correlated with treatment intake. It is possible that the treatment has a bigger (resp. smaller) effect on treated individuals. In that case, ATE is higher (resp. smaller) than TT.

Remark. Another related design is the Brute Force Design among Eligibles. In this design, you impose the treatment status only among eligibles, irrespective of whether they want the treatment or not. It can be operationalized using the selection rule used in Section 3.2.

Example 3.1. Let's use the example to illustrate the concept of ATE. Let's generate data with our usual parameter values without allocating the treatment yet:

```
param <- c(8,.5,.28,1500,0.9,0.01,0.05,0.05,0.05,0.1)
names(param) <- c("barmu","sigma2mu","sigma2U","barY","rho","theta","sigma2epsilon","sigma2eta","delta"
```

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
Ds <- rep(0,N)
Ds[YB<=param["barY"]] <- 1
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
```

In the sample, the ATE is the average difference between y_i^1 and y_i^0, or – the expectation operator being linear – the difference between average y_i^1 and average y_i^0. In our sample, the former is equal to 0.179 and the latter to 0.179.

In the population, the ATE is equal to:

$$\begin{aligned}\Delta^y_{ATE} &= \mathbb{E}[Y_i^1 - Y_i^0] \\ &= \mathbb{E}[\alpha_i] \\ &= \bar{\alpha} + \theta\bar{\mu}.\end{aligned}$$

Let's write a function to compute the value of the ATE and of TT (we derived the formula for TT in the previous lecture):

```
delta.y.ate <- function(param){
  return(param["baralpha"]+param["theta"]*param["barmu"])
}
delta.y.tt <- function(param){
  return(param["baralpha"]+param["theta"]*param["barmu"]-param["theta"]*((param["sigma2mu"]*dnorm((log(
}
```

In the population, with our parameter values, $\Delta^y_{ATE} = 0.18$ and $\Delta^y_{TT} = 0.172$. In our case, selection into the treatment is correlated with lower outcomes, so that $TT \leq ATE$.

In order to implement the Brute Force Design in practice in a sample, we simply either draw a coin repeatedly for each member of the sample, assigning for example, all "heads" to the treatment and all "tails" to the control. Because it can be a little cumbersome, it is possible to replace the coin toss by a pseudo-Random Number Generator (RNG), which is is an algorithm that tries to mimic the properties of random draws. When generating the samples in the numerical exmples, I actually use a pseudo-RNG. For example, we can

draw from a uniform distribution on $[0,1]$ and allocate to the treatment all the individuals whose draw is smaller than 0.5:

$$R_i^* \sim \mathcal{U}[0,1]$$
$$R_i = \begin{cases} 1 & \text{if } R_i^* \leq .5 \\ 0 & \text{if } R_i^* > .5 \end{cases}$$

The advantage of using a uniform law is that you can set up proportions of treated and controls easily.

Example 3.2. In our numerical example, the following R code generates two random groups, one treated and one control, and imposes the Assumption of Brute Force Validity:

```
# randomized allocation of 50% of individuals
Rs <- runif(N)
R <- ifelse(Rs<=.5,1,0)
y <- y1*R+y0*(1-R)
Y <- Y1*R+Y0*(1-R)
```

Remark. It is interesting to stop for one minute to think about how the Brute Force Design solves the FPCI. First, with the ATE, the counterfactual problem is more severe than in the case of the TT. In the routine mode of the program, where only eligible individuals receive the treatment, both parts of the ATE are unobserved:

- $\mathbb{E}[Y_i^1]$ is unobserved since we only observe the expected value of outcomes for the treated $\mathbb{E}[Y_i^1|D_i = 1]$, and they do not have to be the same.
- $\mathbb{E}[Y_i^0]$ is unobserved since we only observe the expected value of outcomes for the untreated $\mathbb{E}[Y_i^0|D_i = 0]$, and they do not have to be the same.

What the Brute Force Design does, is that it allocates randomly one part of the sample to the treatment, so that we see $\mathbb{E}[Y_i^1|R_i = 1] = \mathbb{E}[Y_i^1]$ and one part to the control so that we see $\mathbb{E}[Y_i^0|R_i = 0] = \mathbb{E}[Y_i^0]$.

3.1.2 Estimating ATE

3.1.2.1 Using the WW estimator

In order to estimate ATE in a sample where the treatment has been randomized using a Brute Force Design, we simply use the sample equivalent of the With/Without estimator:

$$\hat{\Delta}_{WW}^Y = \frac{1}{\sum_{i=1}^N R_i} \sum_{i=1}^N Y_i R_i - \frac{1}{\sum_{i=1}^N (1-R_i)} \sum_{i=1}^N Y_i (1-R_i).$$

Example 3.3. In our numerical example, the WW estimator can be computed as follows in the sample:

```
delta.y.ww <- mean(y[R==1])-mean(y[R==0])
```

The WW estimator of the ATE in the sample is equal to 0.156. Let's recall that the true value of ATE is 0.18 in the population and 0.179 in the sample.

We can also see in our example how the Brute Force Design approximates the counterfactual expectation $\mathbb{E}[y_i^1]$ and its sample equivalent mean $\frac{1}{\sum_{i=1}^N} \sum_{i=1}^N y_i^1$ by the observed mean in the treated sample $\frac{1}{\sum_{i=1}^N R_i} \sum_{i=1}^N y_i R_i$. In our example, the sample value of the counterfactual mean potential outcome $\frac{1}{\sum_{i=1}^N} \sum_{i=1}^N y_i^1$ is equal to 8.222 and the sample value of its observed counterpart is 8.209. Similarly, the sample value of the counterfactual mean potential outcome $\frac{1}{\sum_{i=1}^N} \sum_{i=1}^N y_i^0$ is equal to 8.043 and the sample value of its observed counterpart is 8.054.

3.1.2.2 Using OLS

As we have seen in Lecture 0, the WW estimator is numerically identical to the OLS estimator of a linear regression of outcomes on treatment: The OLS coefficient β in the following regression:

$$Y_i = \alpha + \beta R_i + U_i$$

is the WW estimator.
Example 3.4. In our numerical example, we can run the OLS regression as follows:

```
reg.y.R.ols <- lm(y~R)
```

$\hat{\Delta}^y_{OLS} = 0.156$ which is exactly equal, as expected, to the WW estimator: 0.156.

3.1.2.3 Using OLS conditioning on covariates

The advantage of using OLS other the direct WW comparison is that it gives you a direct estimate of sampling noise (see next section) but also that it enables you to condition on additional covariates in the regression: The OLS coefficient β in the following regression:

$$Y_i = \alpha + \beta R_i + \gamma' X_i + U_i$$

is a consistent (and even unbiased) estimate of the ATE.

proof needed, especially assumption of linearity. Also, is interaction between X_i and R_i needed?
Example 3.5. In our numerical example, we can run the OLS regression conditioning on y_i^B as follows:

```
reg.y.R.ols.yB <- lm(y~R + yB)
```

$\hat{\Delta}^y_{OLSX} = 0.177$. Note that $\hat{\Delta}^y_{OLSX} \neq \hat{\Delta}^y_{WW}$. There is no numerical equivalence between the two estimators.
Remark. Why would you want to condition on covariates in an RCT? Indeed, covariates should be balanced by randomization and thus there does not seem to be a rationale for conditioning on potential confounders, since there should be none. The main reason why we condition on covariates is to decrease sampling noise. Remember that sampling noise is due to imbalances between confounders in the treatment and control group. Since these imbalances are not systematic, the WW estimator is unbiased. We can also make the bias due to these unbalances as small as we want by choosing an adequate sample size (the WW estimator is consistent). But for a given sample size, these imbalances generate sampling noise around the true ATE. Conditioning on covariates helps decrease sampling noise by accounting for imbalances due to observed covariates. If observed covariates explain a large part of the variation in outcomes, conditioning on them is going to prevent a lot of sampling noise from occuring.
Example 3.6. In order to make the gains in precision from conditioning on covariates apparent, let's use Monte Carlo simulations of our numerical example.

```
monte.carlo.brute.force.ww <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  Ds <- rep(0,N)
  Ds[YB<=param["barY"]] <- 1
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
```

```
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  # randomized allocation of 50% of individuals
  Rs <- runif(N)
  R <- ifelse(Rs<=.5,1,0)
  y <- y1*R+y0*(1-R)
  Y <- Y1*R+Y0*(1-R)
  reg.y.R.ols <- lm(y~R)
  return(c(reg.y.R.ols$coef[2],sqrt(vcovHC(reg.y.R.ols,type='HC2')[2,2])))
}

simuls.brute.force.ww.N <- function(N,Nsim,param){
  simuls.brute.force.ww <- as.data.frame(matrix(unlist(lapply(1:Nsim,monte.carlo.brute.force.ww,N=N,para
  colnames(simuls.brute.force.ww) <- c('WW','se')
  return(simuls.brute.force.ww)
}

sf.simuls.brute.force.ww.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=2*ncpus)
  sfLibrary(sandwich)
  sim <- as.data.frame(matrix(unlist(sfLapply(1:Nsim,monte.carlo.brute.force.ww,N=N,param=param)),nrow=
  sfStop()
  colnames(sim) <- c('WW','se')
  return(sim)
}

Nsim <- 1000
#Nsim <- 10
N.sample <- c(100,1000,10000,100000)
#N.sample <- c(100,1000,10000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.brute.force.ww <- lapply(N.sample,sf.simuls.brute.force.ww.N,Nsim=Nsim,param=param)
names(simuls.brute.force.ww) <- N.sample
```

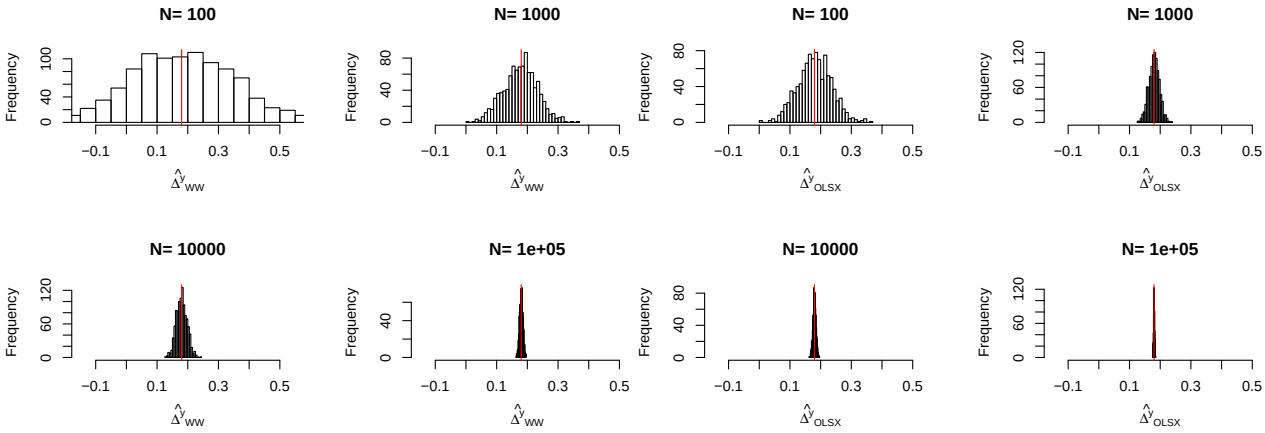
```
monte.carlo.brute.force.ww.yB <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  Ds <- rep(0,N)
  Ds[YB<=param["barY"]] <- 1
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
```

```
  Y0 <- exp(y0)
  Y1 <- exp(y1)
  # randomized allocation of 50% of individuals
  Rs <- runif(N)
  R <- ifelse(Rs<=.5,1,0)
  y <- y1*R+y0*(1-R)
  Y <- Y1*R+Y0*(1-R)
  reg.y.R.yB.ols <- lm(y~R + yB)
  return(c(reg.y.R.yB.ols$coef[2],sqrt(vcovHC(reg.y.R.yB.ols,type='HC2')[2,2])))
}

simuls.brute.force.ww.yB.N <- function(N,Nsim,param){
  simuls.brute.force.ww.yB <- as.data.frame(matrix(unlist(lapply(1:Nsim,monte.carlo.brute.force.ww.yB,N
  colnames(simuls.brute.force.ww.yB) <- c('WW','se')
  return(simuls.brute.force.ww.yB)
}

sf.simuls.brute.force.ww.yB.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=2*ncpus)
  sfLibrary(sandwich)
  sim <- as.data.frame(matrix(unlist(sfLapply(1:Nsim,monte.carlo.brute.force.ww.yB,N=N,param=param)),nr
  sfStop()
  colnames(sim) <- c('WW','se')
  return(sim)
}

Nsim <- 1000
#Nsim <- 10
N.sample <- c(100,1000,10000,100000)
#N.sample <- c(100,1000,10000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.brute.force.ww.yB <- lapply(N.sample,sf.simuls.brute.force.ww.yB.N,Nsim=Nsim,param=param)
names(simuls.brute.force.ww.yB) <- N.sample
```

Figure 3.1 shows that the gains in precision from conditioning on y_i^B are spectacular in our numerical example. They basically correspond to a gain in one order of magnitude of sample size: the precision of the $OLSX$ estimator conditioning on y_i^B with a sample size of 100 is similar to the precision of the OLS estimator not conditioning on y_i^B with a sample size of 1000. This large gain in precision is largely due to the fact that y_i and y_i^B are highly correlated. Not all covariates perform so well in actual samples in the social sciences.

Remark. The ability to condition on covariates in order to decrease sampling noise is a blessing but can also be a curse when combined with significance testing. Indeed, you can now see that you can run a lot of regressions (with and without some covariates, interactions, etc) and maybe report only the statistically significant ones. This is a bad practice that will lead to publication bias and inflated treatment effects. Several possibilities in order to avoid that:

1. Pre-register your analysis and explain which covariates you are going to use (with which interactions, etc) so that you cannot cherry pick your favorite results ex-post.
2. Use a stratified design for your RCT (more on this in Lecture 6) so that the important covariates are already balanced between treated and controls.
3. If unable to do all of the above, report results from regressions without controls and with various sets of controls. We do not expect the various treatment effect estimates to be the same (they cannot be, otherwise, they would have similar sampling noise), but we expect the following pattern: conditioning



(a) *WW* (b) OLSX

Figure 3.1: Distribution of the *WW* and *OLSX* estimators in a Brute Force design over replications of samples of different sizes

should systematically decrease sampling noise, not increase the treatment effect estimate. If conditioning on covariates makes a treatment effect significant, pay attention to why: is it because of a decrease in sampling noise (expected and OK) or because of an increase in treatment effect (beware specification search).

Revise that especially in light of Chapter 12

Remark. You might not be happy with the assumption of linearity needed to use OLS to control for covariates. I have read somewhere (forgot where) that this should not be much of a problem since covariates are well balanced between groups by randomization, and thus a linear first approximation to the function relating X_i to Y_i should be fine. I tend not to buy that argument much. I have to run simulations with a non linear relation between outcomes and controls and see how linear OLS performs. If you do not like the linearity assumption, you can always use any of the nonparametric observational methods presented in Chapter 5.

3.1.2.4 Estimating Sampling Noise

In order to estimate sampling noise, you can either use the CLT-based approach or resampling, either using the bootstrap or randomization inference. In Section 2.2, we have already discussed how to estimate sampling noise when using the WW estimator that we are using here. We are going to use the default and heteroskedasticity-robust standard errors from OLS, which are both CLT-based. Only the heteroskedasticity-robust standard errors are valid under the assumptions that we have made so far. Homoskedasticity would require constant treatment effects. Heteroskedasticity being small in our numerical example, that should not matter much, but it could in other applications.

Example 3.7. Let us first estimate sampling noise for the simple WW estimator without control variables, using the OLS estimator.

```
sn.BF.simuls <- 2*quantile(abs(simuls.brute.force.ww[['1000']][,'WW']-delta.y.ate(param)),probs=c(0.99))
sn.BF.OLS.hetero <- 2*qnorm((delta+1)/2)*sqrt(vcovHC(reg.y.R.ols,type='HC2')[2,2])
sn.BF.OLS.homo <- 2*qnorm((delta+1)/2)*sqrt(vcov(reg.y.R.ols)[2,2])
```

The true value of the 99% sampling noise with a sample size of 1000 and no control variables is stemming from the simulations is 0.274. The 99% sampling noise estimated using heteroskedasticity robust OLS standard errors is 0.295. The 99% sampling noise estimated using default OLS standard errors is 0.294.

Let us now estimate sampling noise for the simple WW estimator conditioning on y_i^B, using the OLS estimator.

```
sn.BF.simuls.yB <- 2*quantile(abs(simuls.brute.force.ww.yB[['1000']][,'WW']-delta.y.ate(param)),probs=c
sn.BF.OLS.hetero.yB <- 2*qnorm((delta+1)/2)*sqrt(vcovHC(reg.y.R.ols.yB,type='HC2')[2,2])
sn.BF.OLS.homo.yB <- 2*qnorm((delta+1)/2)*sqrt(vcov(reg.y.R.ols.yB)[2,2])
```

The true value of the 99% sampling noise with a sample size of 1000 and no control variables is stemming from the simulations is 0.088. The 99% sampling noise estimated using heteroskedasticity robust OLS standard errors is 0.092. The 99% sampling noise estimated using default OLS standard errors is 0.091.

Let's see how all of this works on average. Figure 3.2 shows that overall the sampling nois eis much lower with $OLSX$ than with WW, as expected from Figure 3.1. The CLT-based estimator of sampling noise accounting for heteroskedasticity (in blue) recovers true sampling noise (in red) pretty well. Figure 3.3 shows that the CLT-based estimates of sampling noise are on point, except for $N = 10000$, where the CLT slightly overestimates true sampling noise. Figure 3.4 shows what happens when conditioning on Y^B in a selection of 40 samples. The reduction in samplong noise is pretty drastic here.

```
for (k in (1:length(N.sample))){
  simuls.brute.force.ww[[k]]$CLT.noise <- 2*qnorm((delta+1)/2)*simuls.brute.force.ww[[k]][,'se']
  simuls.brute.force.ww.yB[[k]]$CLT.noise <- 2*qnorm((delta+1)/2)*simuls.brute.force.ww.yB[[k]][,'se']
}

samp.noise.ww.BF <- sapply(lapply(simuls.brute.force.ww,`[`,,1),samp.noise,delta=delta)
precision.ww.BF <- sapply(lapply(simuls.brute.force.ww,`[`,,1),precision,delta=delta)
names(precision.ww.BF) <- N.sample
signal.to.noise.ww.BF <- sapply(lapply(simuls.brute.force.ww,`[`,,1),signal.to.noise,delta=delta,param=p
names(signal.to.noise.ww.BF) <- N.sample
table.noise.BF <- cbind(samp.noise.ww.BF,precision.ww.BF,signal.to.noise.ww.BF)
colnames(table.noise.BF) <- c('sampling.noise', 'precision', 'signal.to.noise')
table.noise.BF <- as.data.frame(table.noise.BF)
table.noise.BF$N <- as.numeric(N.sample)
table.noise.BF$ATE <- rep(delta.y.ate(param),nrow(table.noise.BF))
for (k in (1:length(N.sample))){
  table.noise.BF$CLT.noise[k] <- mean(simuls.brute.force.ww[[k]]$CLT.noise)
}
table.noise.BF$Method <- rep("WW",nrow(table.noise.BF))

samp.noise.ww.BF.yB <- sapply(lapply(simuls.brute.force.ww.yB,`[`,,1),samp.noise,delta=delta)
precision.ww.BF.yB <- sapply(lapply(simuls.brute.force.ww.yB,`[`,,1),precision,delta=delta)
names(precision.ww.BF.yB) <- N.sample
signal.to.noise.ww.BF.yB <- sapply(lapply(simuls.brute.force.ww.yB,`[`,,1),signal.to.noise,delta=delta,p
names(signal.to.noise.ww.BF.yB) <- N.sample
table.noise.BF.yB <- cbind(samp.noise.ww.BF.yB,precision.ww.BF.yB,signal.to.noise.ww.BF.yB)
colnames(table.noise.BF.yB) <- c('sampling.noise', 'precision', 'signal.to.noise')
table.noise.BF.yB <- as.data.frame(table.noise.BF.yB)
table.noise.BF.yB$N <- as.numeric(N.sample)
table.noise.BF.yB$ATE <- rep(delta.y.ate(param),nrow(table.noise.BF.yB))
for (k in (1:length(N.sample))){
  table.noise.BF.yB$CLT.noise[k] <- mean(simuls.brute.force.ww.yB[[k]]$CLT.noise)
}
table.noise.BF.yB$Method <- rep("OLSX",nrow(table.noise.BF))

table.noise.BF.tot <- rbind(table.noise.BF,table.noise.BF.yB)
table.noise.BF.tot$Method <- factor(table.noise.BF.tot$Method,levels=c("WW","OLSX"))

ggplot(table.noise.BF.tot, aes(x=as.factor(N), y=ATE,fill=Method)) +
  geom_bar(position=position_dodge(), stat="identity", colour='black') +
```

```
  geom_errorbar(aes(ymin=ATE-sampling.noise/2, ymax=ATE+sampling.noise/2), width=.2,position=position_d
  geom_errorbar(aes(ymin=ATE-CLT.noise/2, ymax=ATE+CLT.noise/2), width=.2,position=position_dodge(.9),c
  xlab("Sample Size")+
  theme_bw()+
  theme(legend.position=c(0.85,0.88))
```

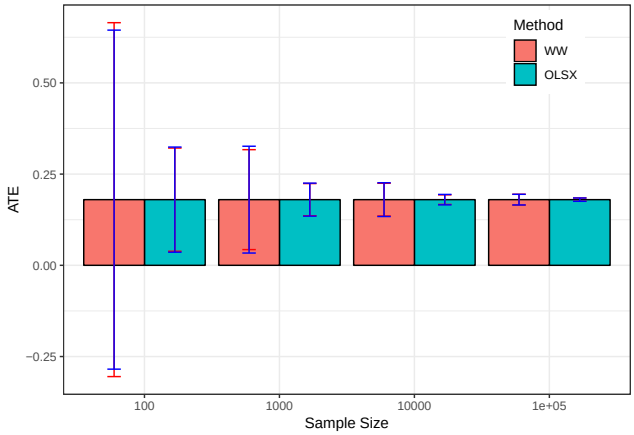
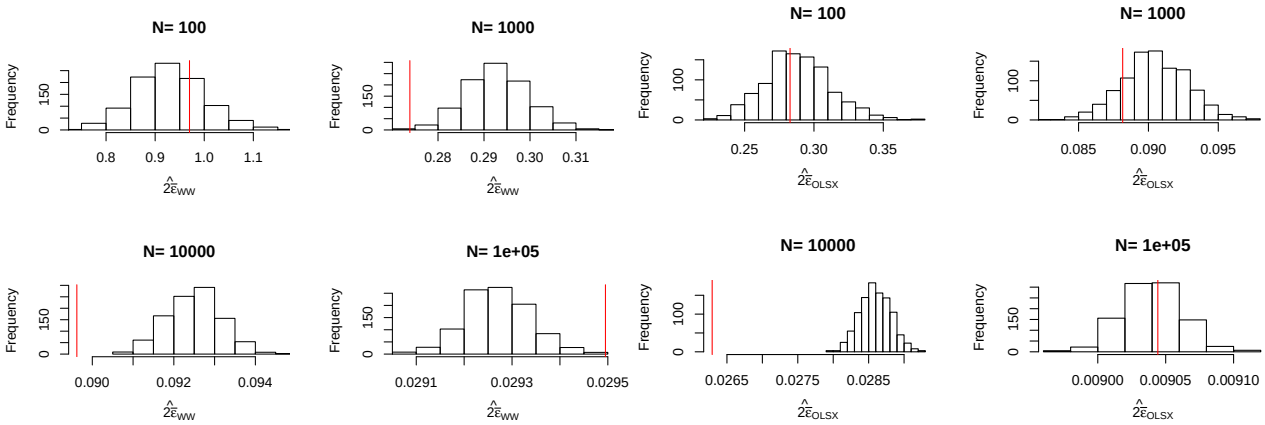


Figure 3.2: Average CLT-based approximations of sampling noise in the Brute Force design for WW and $OLSX$ over replications of samples of different sizes (true sampling noise in red)

```
par(mfrow=c(2,2))
for (i in 1:length(simuls.brute.force.ww)){
  hist(simuls.brute.force.ww[[i]][,'CLT.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expressi
  abline(v=table.noise.BF[i,colnames(table.noise)=='sampling.noise'],col="red")
}
par(mfrow=c(2,2))
for (i in 1:length(simuls.brute.force.ww.yB)){
  hist(simuls.brute.force.ww.yB[[i]][,'CLT.noise'],main=paste('N=',as.character(N.sample[i])),xlab=expre
  abline(v=table.noise.BF.yB[i,colnames(table.noise)=='sampling.noise'],col="red")
}
```



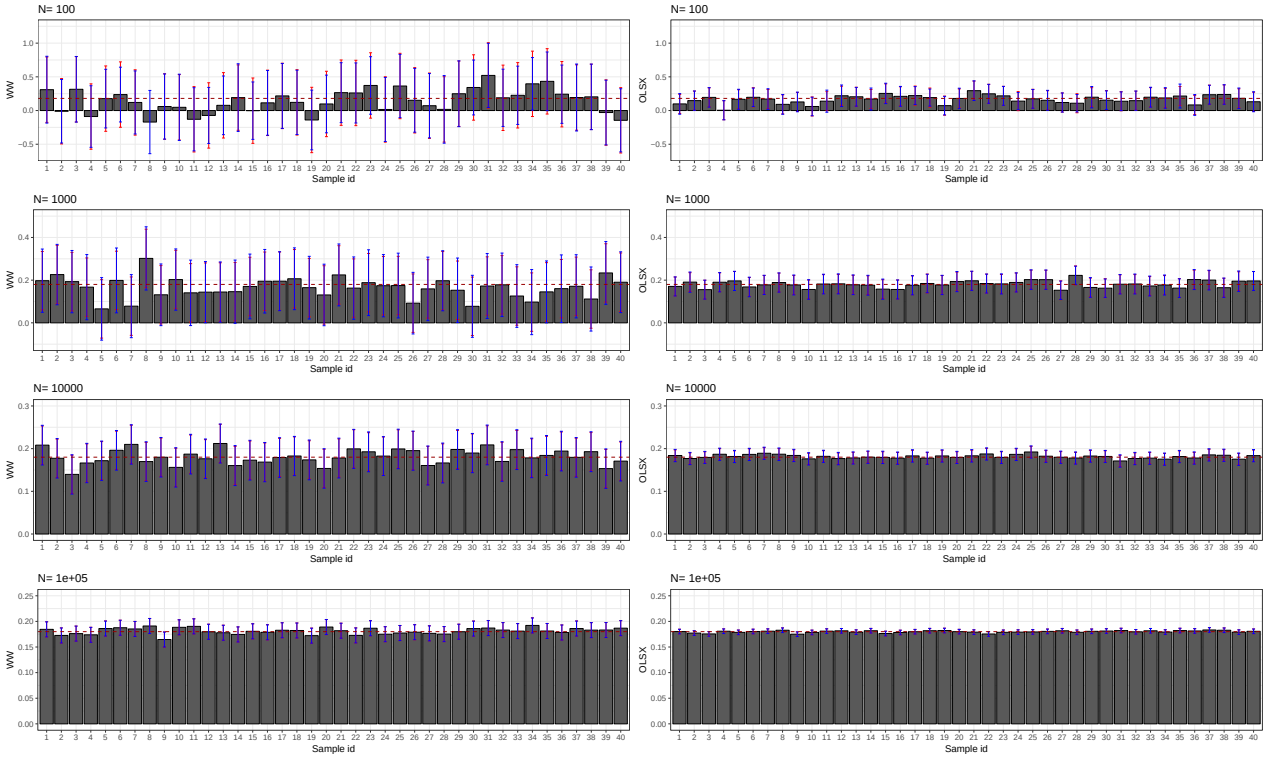
(a) WW (b) OLSX

Figure 3.3: Distribution of the CLT approximation of sampling noise in the Brute Force design for WW and $OLSX$ over replications of samples of different sizes (true sampling noise in red)

```
N.plot <- 40
plot.list <- list()
limx <- list(c(-0.65,1.25),c(-0.1,0.5),c(0,0.30),c(0,0.25))

for (k in 1:length(N.sample)){
  set.seed(1234)
  test.CLT.BF <- simuls.brute.force.ww[[k]][sample(N.plot),c('WW','CLT.noise')]
  test.CLT.BF <- as.data.frame(cbind(test.CLT.BF,rep(samp.noise(simuls.brute.force.ww[[k]][,'WW'],delta=
  colnames(test.CLT.BF) <- c('WW','CLT.noise','sampling.noise')
  test.CLT.BF$id <- 1:N.plot
  plot.test.CLT.BF <- ggplot(test.CLT.BF, aes(x=as.factor(id), y=WW)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
      geom_errorbar(aes(ymin=WW-CLT.noise/2, ymax=WW+CLT.noise/2), width=.2,position=position_dodge(.9)
      geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
      ylim(limx[[k]][1],limx[[k]][2])+
      xlab("Sample id")+
      theme_bw()+
      ggtitle(paste("N=",N.sample[k]))
  plot.list[[k]] <- plot.test.CLT.BF
}
plot.CI.BF <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=length(
print(plot.CI.BF)

plot.list <- list()
for (k in 1:length(N.sample)){
  set.seed(1234)
  test.CLT.BF.yB <- simuls.brute.force.ww.yB[[k]][sample(N.plot),c('WW','CLT.noise')]
  test.CLT.BF.yB <- as.data.frame(cbind(test.CLT.BF.yB,rep(samp.noise(simuls.brute.force.ww.yB[[k]][,'W
  colnames(test.CLT.BF.yB) <- c('WW','CLT.noise','sampling.noise')
  test.CLT.BF.yB$id <- 1:N.plot
  plot.test.CLT.BF.yB <- ggplot(test.CLT.BF.yB, aes(x=as.factor(id), y=WW)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=WW-sampling.noise/2, ymax=WW+sampling.noise/2), width=.2,position=position_
      geom_errorbar(aes(ymin=WW-CLT.noise/2, ymax=WW+CLT.noise/2), width=.2,position=position_dodge(.9)
      geom_hline(aes(yintercept=delta.y.ate(param)), colour="#990000", linetype="dashed")+
      ylim(limx[[k]][1],limx[[k]][2])+
      xlab("Sample id")+
      ylab("OLSX")+
      theme_bw()+
      ggtitle(paste("N=",N.sample[k]))
  plot.list[[k]] <- plot.test.CLT.BF.yB
}
plot.CI.BF.yB <- plot_grid(plot.list[[1]],plot.list[[2]],plot.list[[3]],plot.list[[4]],ncol=1,nrow=leng
print(plot.CI.BF.yB)
```

3.2 Randomization After Self-Selection

In Randomization After Self-Selection, individuals are randomly assigned to the treatment after having expressed their willingness to receive it. This design is able to recover the average effect of the Treatment on the Treated (TT).



(a) WW (b) OLSX

Figure 3.4: CLT-based confidence intervals of $\hat{WW}$ and $\hat{OLSX}$ for $\delta = 0.99$ over sample replications for various sample sizes (true confidence intervals in red)

In order to explain this design clearly, and especially to make it clear how it differs from the following one (randomization after eligibility), I have to introduce a slightly more complex selection rule that we have seen so far, one that includes self-selection, *i.e.* take-up decisions by agents. We are going to assume that there are two steps in agents' participation process:

- Eligibility: agents' eligibility is assessed first, giving rise to a group of eligible individuals ($E_i = 1$) and a group of non eligible individuals ($E_i = 0$).
- Self-selection: eligible agents can then decide whether they want to take-up the proposed treatment or not. $D_i = 1$ for those who do. $D_i = 0$ for those who do not. By convention, ineligibles have $D_i = 0$.

Example 3.8. In our numerical example, here are the equations operationalizing these notions:

$$\begin{aligned} E_i &= \mathbb{1}[y_i^B \leq \bar{y}] \\ D_i &= \mathbb{1}[\underbrace{\bar{\alpha} + \theta\bar{\mu} - C_i}_{D_i^*} \geq 0 \wedge E_i = 1] \\ C_i &= \bar{c} + \gamma\mu_i + V_i \\ V_i &\sim \mathcal{N}(0, \sigma_V^2) \end{aligned}$$

Eligibility is still decided based on pre-treatment outcomes being smaller than a threshold level $\bar{y}$. Self-selection among eligibles is decided by the net utility of the treatment D_i^* being positive. Here, the net utility is composed of the average gain from the treatment (assuming agents cannot foresee their idiosyncratic gain from the treatment apart from the part depending on μ_i) $\bar{\alpha} + \theta\bar{\mu}$ minus the cost of participation C_i. The cost of participation in turn depends on a constant, on μ_i and on a random shock orthogonal to everything else V_i. This cost might represent the administrative cost of applying for the treatment and the opportunity cost of participating into the treatment (foregone earnings and/or cost of time). Conditional on eligiblity, self-selection is endogenous in this model since both the gains and the cost of participation depend on μ_i. Costs depend on μ_i since most productive people may face lower administrative costs but a higher opportunity cost of time.

Let's choose some values for the new parameters:

```
param <- c(param,-6.25,0.9,0.5)
names(param) <- c("barmu","sigma2mu","sigma2U","barY","rho","theta","sigma2epsilon","sigma2eta","delta"
```

and let's generate a new dataset:

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
E <- ifelse(YB<=param["barY"],1,0)
V <- rnorm(N,0,param["sigma2V"])
Dstar <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
Ds <- ifelse(Dstar>=0 & E==1,1,0)
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
```

Let's compute the value of the TT parameter in this new model:

$$\Delta^y_{TT} = \bar{\alpha} + \theta\mathbb{E}[\mu_i | \mu_i + U^B_i \leq \bar{y} \wedge \bar{\alpha} + \theta\bar{\mu} - \bar{c} - \gamma\mu_i - V_i \geq 0]$$

To compute the expectation of a doubly censored normal, I use the package `tmvtnorm`.

$$(\mu_i, y^B_i, D^*_i) = \mathcal{N}\left(\bar{\mu}, \bar{\mu}, \bar{\alpha} + (\theta - \gamma)\bar{\mu} - \bar{c}, \begin{pmatrix} \sigma^2_\mu & \sigma^2_\mu & -\gamma\sigma^2_\mu \\ \sigma^2_\mu & \sigma^2_\mu + \sigma^2_U & -\gamma\sigma^2_\mu \\ -\gamma\sigma^2_\mu & -\gamma\sigma^2_\mu & \gamma^2\sigma^2_\mu + \sigma^2_V \end{pmatrix}\right)$$

```
mean.mu.yB.Dstar <- c(param['barmu'],param['barmu'],param['baralpha']- param['barc']+(param['theta']-pa
cov.mu.yB.Dstar <- matrix(c(param['sigma2mu'],param["sigma2mu"],-param['gamma']*param["sigma2mu"],
                            param["sigma2mu"],param['sigma2mu']+param['sigma2U'],-param['gamma']*param[
                            -param['gamma']*param["sigma2mu"],-param['gamma']*param["sigma2mu"],param["
lower.cut <- c(-Inf,-Inf,0)
upper.cut <- c(Inf,log(param['barY']),Inf)
moments.cut <- mtmvnorm(mean=mean.mu.yB.Dstar,sigma=cov.mu.yB.Dstar,lower=lower.cut,upper=upper.cut)
delta.y.tt <- param['baralpha']+ param['theta']*moments.cut$tmean[1]
delta.y.ww.self.select <- mean(y[R==1 & Ds==1])-mean(y[R==0 & Ds==1])
```

The value of Δ^y_{TT} in our illustration is now 0.17.

3.2.1 Identification

With Randomization After Self-Selection, identification requires two assumptions:
Definition 3.3 (Independence Among Self-Selected)**.** We assume that the randomized allocation of the program among applicants is well done:

$$R_i \perp\!\!\!\perp (Y^0_i, Y^1_i) | D_i = 1.$$

Independence can be enforced by the randomized allocation of the treatment among the eligible applicants.

We need a second assumption:
Definition 3.4 (Randomization After Self-Selection Validity)**.** We assume that the randomized allocation of the program does not interfere with how potential outcomes and self-selection are generated:

$$Y_i = \begin{cases} Y^1_i & \text{if } (R_i = 1 \text{ and } D_i = 1) \\ Y^0_i & \text{if } (R_i = 0 \text{ and } D_i = 1) \text{ or } D_i = 0 \end{cases}$$

with Y^1_i, Y^0_i and D_i the same potential outcomes and self-selection decisions as in a routine allocation of the treatment.

Under these assumptions, we have the following result:
Theorem 3.2 (Identification With Randomization After Self-Selection)**.** *Under Assumptions 3.3 and 3.4, the WW estimator among the self-selected identifies TT:*

$$\Delta^Y_{WW|D=1} = \Delta^Y_{TT},$$

with:

$$\Delta^Y_{WW|D=1} = \mathbb{E}[Y_i|R_i = 1, D_i = 1] - \mathbb{E}[Y_i|R_i = 0, D_i = 1].$$

Proof.

$$\begin{aligned}\Delta^Y_{WW|D=1} &= \mathbb{E}[Y_i|R_i = 1, D_i = 1] - \mathbb{E}[Y_i|R_i = 0, D_i = 1] \\ &= \mathbb{E}[Y_i^1|R_i = 1, D_i = 1] - \mathbb{E}[Y_i^0|R_i = 0, D_i = 1] \\ &= \mathbb{E}[Y_i^1|D_i = 1] - \mathbb{E}[Y_i^0|D_i = 1] \\ &= \mathbb{E}[Y_i^1 - Y_i^0|D_i = 1],\end{aligned}$$

where the second equality uses Randomization After Self-Selection Validity, the third equality Independence Among Self-Selected and the last equality the linearity of the expectation operator. □

Remark. The key intuitions for how Randomization After Self-Selection solves the FPCI are:

- By allowing for eligibilty and self-selection, we identify the agents that would benefit from the treatment in routine mode (the treated).
- By randomly denying the treatment to some of the treated, we can estimate the counterfactual outcome of the treated by looking at the counterfactual outcome of the denied applicants: $\mathbb{E}[Y_i^0|D_i = 1] = \mathbb{E}[Y_i|R_i = 0, D_i = 1]$.

Remark. In practice, we use a pseudo-RNG to generate a random allocation among applicants:

$$\begin{aligned}R_i^* &\sim \mathcal{U}[0, 1] \\ R_i &= \begin{cases} 1 & \text{if } R_i^* \leq .5 \wedge D_i = 1 \\ 0 & \text{if } R_i^* > .5 \wedge D_i = 1 \end{cases}\end{aligned}$$

Example 3.9. In our numerical example, the following R code generates two random groups, one treated and one control, and imposes the Assumption of Randomization After Self-Selection Validity:

```
#random allocation among self-selected
Rs <- runif(N)
R <- ifelse(Rs<=.5 & Ds==1,1,0)
y <- y1*R+y0*(1-R)
Y <- Y1*R+Y0*(1-R)
```

3.2.2 Estimating TT

3.2.2.1 Using the WW Estimator

As in the case of the Brute Force Design, we can use the WW estimator to estimate the effect of the program with Randomization After Self-Selection, except that this time the WW estimator is applied among applicant to the program only:

$$\hat{\Delta}^Y_{WW|D=1} = \frac{1}{\sum_{i=1}^N D_i R_i} \sum_{i=1}^N Y_i D_i R_i - \frac{1}{\sum_{i=1}^N D_i(1 - R_i)} \sum_{i=1}^N D_i Y_i (1 - R_i).$$

Example 3.10. In our numerical example, we can form the WW estimator among applicants:

```
delta.y.ww.self.select <- mean(y[R==1 & Ds==1])-mean(y[R==0 & Ds==1])
```

WW among applicants is equal to 0.085. It is actually rather far from the true value of 0.17, which reminds us that unbiasedness does not mean that a given sample will not suffer from a large bias. We just drew a bad sample where confounders are not very well balanced.

3.2.2.2 Using OLS

As in the Brute Force Design with the ATE, we can estimate the TT parameter with Randomization After Self-Selection using the OLS estimator. In the following regression run among applicants only (with $D_i = 1$), β estimates TT:

$$Y_i = \alpha + \beta R_i + U_i.$$

As a matter of fact, the OLS estimator without control variables is numerically equivalent to the WW estimator.

Example 3.11. In our numerical example, here is the OLS regression:

```
reg.y.R.ols.self.select <- lm(y[Ds==1]~R[Ds==1])
```

The value of the OLS estimator is 0.085, which is identical to the WW estimator among applicants.

3.2.2.3 Using OLS Conditioning on Covariates

We might want to condition on covariates in order to reduce the amount of sampling noise. Parametrically, we can run the following OLS regression among applicants (with $D_i = 1$):

$$Y_i = \alpha + \beta R_i + \gamma' X_i + U_i.$$

β estimates the TT.

Needed: proof. Especially check whether we need to center covariates at the mean of the treatment group. I think so.

We can also use Matching to obtain a nonparametric estimator.

Example 3.12. Let us first compute the OLS estimator conditioning on y_i^B:

```
reg.y.R.yB.ols.self.select <- lm(y[Ds==1] ~ R[Ds==1] + yB[Ds==1])
```

Our estimate of TT after conditioning on y_i^B is 0.145. Conditioning on y_i^B has been able to solve part of the bias of the WW problem estimator.

Let's now check whether conditioning on OLS has brought an improvement in terms of decreased sampling noise.

```
monte.carlo.self.select.ww <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  E <- ifelse(YB<=param["barY"],1,0)
  V <- rnorm(N,0,param["sigma2V"])
```

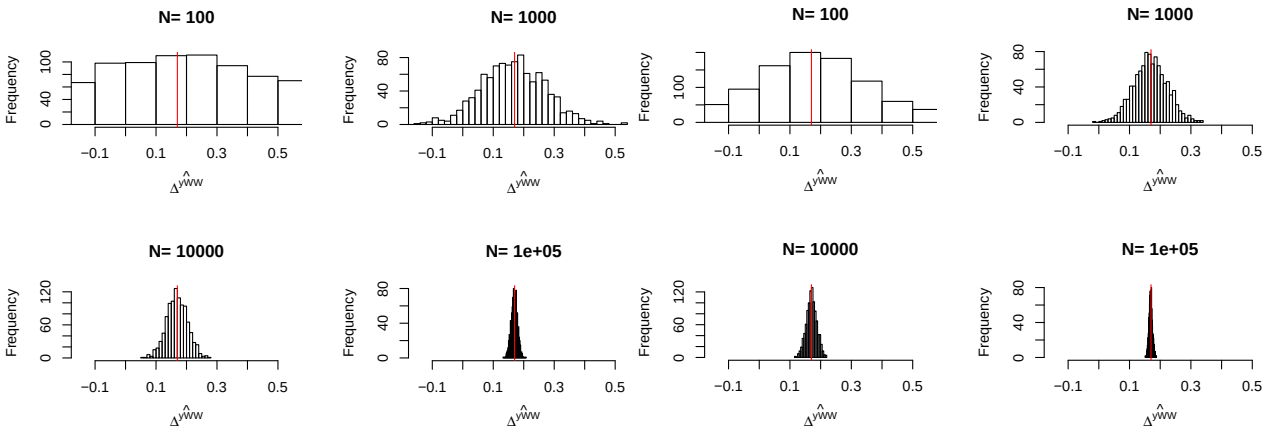
```
  Dstar <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
  Ds <- ifelse(Dstar>=0 & E==1,1,0)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)

  #random allocation among self-selected
  Rs <- runif(N)
  R <- ifelse(Rs<=.5 & Ds==1,1,0)
  y <- y1*R+y0*(1-R)
  Y <- Y1*R+Y0*(1-R)
  return(mean(y[R==1 & Ds==1])-mean(y[R==0 & Ds==1]))
}

simuls.self.select.ww.N <- function(N,Nsim,param){
  simuls.self.select.ww <- matrix(unlist(lapply(1:Nsim,monte.carlo.self.select.ww,N=N,param=param)),nrow
  colnames(simuls.self.select.ww) <- c('WW')
  return(simuls.self.select.ww)
}

sf.simuls.self.select.ww.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=8)
  sim <- matrix(unlist(sfLapply(1:Nsim,monte.carlo.self.select.ww,N=N,param=param)),nrow=Nsim,ncol=1,by
  sfStop()
  colnames(sim) <- c('WW')
  return(sim)
}

Nsim <- 1000
#Nsim <- 10
N.sample <- c(100,1000,10000,100000)
#N.sample <- c(100,1000,10000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.self.select.ww <- lapply(N.sample,sf.simuls.self.select.ww.N,Nsim=Nsim,param=param)
names(simuls.self.select.ww) <- N.sample
```

```
monte.carlo.self.select.yB.ww <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  E <- ifelse(YB<=param["barY"],1,0)
  V <- rnorm(N,0,param["sigma2V"])
  Dstar <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
  Ds <- ifelse(Dstar>=0 & E==1,1,0)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
```

```
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)

  #random allocation among self-selected
  Rs <- runif(N)
  R <- ifelse(Rs<=.5 & Ds==1,1,0)
  y <- y1*R+y0*(1-R)
  Y <- Y1*R+Y0*(1-R)
  reg.y.R.yB.ols.self.select <- lm(y[Ds==1] ~ R[Ds==1] + yB[Ds==1])
  return(reg.y.R.yB.ols.self.select$coef[2])
}

simuls.self.select.yB.ww.N <- function(N,Nsim,param){
  simuls.self.select.yB.ww <- matrix(unlist(lapply(1:Nsim,monte.carlo.self.select.yB.ww,N=N,param=param
  colnames(simuls.self.select.yB.ww) <- c('WW')
  return(simuls.self.select.yB.ww)
}

sf.simuls.self.select.yB.ww.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=8)
  sim <- matrix(unlist(sfLapply(1:Nsim,monte.carlo.self.select.yB.ww,N=N,param=param)),nrow=Nsim,ncol=1
  sfStop()
  colnames(sim) <- c('WW')
  return(sim)
}

Nsim <- 1000
#Nsim <- 10
N.sample <- c(100,1000,10000,100000)
#N.sample <- c(100,1000,10000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.self.select.yB.ww <- lapply(N.sample,sf.simuls.self.select.yB.ww.N,Nsim=Nsim,param=param)
names(simuls.self.select.yB.ww) <- N.sample
```

```
par(mfrow=c(2,2))
for (i in 1:length(simuls.self.select.ww)){
  hist(simuls.self.select.ww[[i]][,'WW'],breaks=30,main=paste('N=',as.character(N.sample[i])),xlab=expr
  abline(v=delta.y.tt,col="red")
}
par(mfrow=c(2,2))
for (i in 1:length(simuls.self.select.yB.ww)){
  hist(simuls.self.select.yB.ww[[i]][,'WW'],breaks=30,main=paste('N=',as.character(N.sample[i])),xlab=e
  abline(v=delta.y.tt,col="red")
}
```

Figure 3.5 shows that, in our example, conditioning on covariates improves precision by the same amount as an increase in sample size by almost one order of magnitude.



(a) WW (b) $OLSX$

Figure 3.5: Distribution of the WW and $OLSX$ estimator with randomization after self-selection over replications of samples of different sizes

3.2.3 Estimating Sampling Noise

In order to estimate precision, we can either use the CLT, deriving sampling noise from the heteroskedasticity-robust standard error OLS estimates, or we can use some form of resampling as the bootstrap or randomization inference.

Example 3.13. Let us derive the CLT-based estimates of sampling noise using the OLS standard errors without conditioning on covariates first. I'm using the sample size with $N = 1000$ as an example.

```
sn.RASS.simuls <- 2*quantile(abs(simuls.self.select.ww[['1000']][,'WW']-delta.y.tt),probs=c(0.99))
sn.RASS.OLS.homo <- 2*qnorm((.99+1)/2)*sqrt(vcov(reg.y.R.ols.self.select)[2,2])
sn.RASS.OLS.hetero <- 2*qnorm((.99+1)/2)*sqrt(vcovHC(reg.y.R.ols.self.select,type='HC2')[2,2])
```

True 99% sampling noise (from the simulations) is 0.548. 99% sampling noise estimated using default OLS standard errors is 0.578. 99% sampling noise estimated using heteroskedasticity robust OLS standard errors is 0.58.

Conditioning on covariates:

```
sn.RASS.simuls.yB <- 2*quantile(abs(simuls.self.select.yB.ww[['1000']][,'WW']-delta.y.tt),probs=c(0.99))
sn.RASS.OLS.homo.yB <- 2*qnorm((.99+1)/2)*sqrt(vcov(reg.y.R.yB.ols.self.select)[2,2])
sn.RASS.OLS.hetero.yB <- 2*qnorm((.99+1)/2)*sqrt(vcovHC(reg.y.R.yB.ols.self.select,type='HC2')[2,2])
```

True 99% sampling noise (from the simulations) is 0.295. 99% sampling noise estimated using default OLS standard errors is 0.294. 99% sampling noise estimated using heteroskedasticity robust OLS standard errors is 0.299.

3.3 Randomization After Eligibility

In Randomization After Eligiblity, we randomly select two groups among the eligibles. Members of the treated group are informed that they are eligible to the program and are free to self-select into it. Members of the control group are not enformed that they are eligible and cannot enroll into the program. With Randomization After Eligiblity, we can still recover the TT despite the fact that we have not randomized access to the programs among the applicants. This is the magic of instrumental variables. Let us detail the mechanics of this beautiful result.

3.3.1 Identification

In order to state the identification results in the Randomization After Eligibility design rigorously, I need to define new potential outcomes:

- $Y_i^{d,r}$ is the value of the outcome Y when individual i belongs to the program group d ($d \in \{0,1\}$) and has been randomized in group r ($r \in \{0,1\}$).
- D_i^r is the value of the program participation decision when individual i has been assigned randomly to group r.

3.3.1.1 Identification of TT

In a Randomization After Eligiblity, we need three assumptions to ensure identification of the TT:
Definition 3.5 (Independence Among Eligibles)**.** We assume that the randomized allocation of the program among eligibles is well done:

$$R_i \perp\!\!\!\perp (Y_i^{0,0}, Y_i^{0,1}, Y_i^{1,0}, Y_i^{1,1}, D_i^1, D_i^0) | E_i = 1.$$

Independence can be enforced by the randomized allocation of information about eligibility among the eligibles.

We need a second assumption:
Definition 3.6 (Randomization After Eligibility Validity)**.** We assume that no eligibles that has been randomized out can take the treatment and that the randomized allocation of the program does not interfere with how potential outcomes and self-selection are generated:

$$\begin{aligned} D_i^0 &= 0, \forall i, \\ D_i &= D_i^1 R_i + (1 - R_i) D_i^0 \\ Y_i &= \begin{cases} Y_i^{1,1} & \text{if } (R_i = 1 \text{ and } D_i = 1) \\ Y_i^{0,1} & \text{if } (R_i = 1 \text{ and } D_i = 0) \\ Y_i^{0,0} & \text{if } R_i = 0 \end{cases} \end{aligned}$$

with $Y_i^{1,1}$, $Y_i^{0,1}$, $Y_i^{0,0}$, D_i^1 and D_i^0 the same potential outcomes and self-selection decisions as in a routine allocation of the treatment.

We need a third assumption:
Definition 3.7 (Exclusion Restriction of Eligibility)**.** We assume that there is no direct effect of being informed about eligibliity to the program on outcomes:

$$\begin{aligned} Y_i^{1,1} &= Y_i^{1,0} = Y_i^1 \\ Y_i^{0,1} &= Y_i^{0,0} = Y_i^0. \end{aligned}$$

Under these assumptions, we have the following result:
Theorem 3.3 (Identification of TT With Randomization After Eligibility)**.** *Under Assumptions 3.5, 3.6 and 3.7, the Bloom estimator among eligibles identifies TT:*

$$\Delta_{Bloom|E=1}^Y = \Delta_{TT}^Y,$$

with:

$$\Delta^Y_{Bloom|E=1} = \frac{\Delta^Y_{WW|E=1}}{\Pr(D_i = 1|R_i = 1, E_i = 1)}$$
$$\Delta^Y_{WW|E=1} = \mathbb{E}[Y_i|R_i = 1, E_i = 1] - \mathbb{E}[Y_i|R_i = 0, E_i = 1].$$

Proof. I keep the conditioning on $E_i = 1$ implicit all along to save notation.

$$\begin{aligned}
\mathbb{E}[Y_i|R_i = 1] &= \mathbb{E}[Y_i^{1,1}D_i + Y_i^{0,1}(1 - D_i)|R_i = 1] \\
&= \mathbb{E}[Y_i^0 + D_i(Y_i^1 - Y_i^0)|R_i = 1] \\
&= \mathbb{E}[Y_i^0|R_i = 1] + \mathbb{E}[Y_i^1 - Y_i^0|D_i = 1, R_i = 1] \Pr(D_i = 1|R_i = 1) \\
&= \mathbb{E}[Y_i^0] + \mathbb{E}[Y_i^1 - Y_i^0|D_i = 1] \Pr(D_i = 1|R_i = 1),
\end{aligned}$$

where the first equality uses Assumption 3.6, the second equality Assumption 3.7 and the last equality Assumption 3.5 and the fact that $D_i = 1 \Rightarrow R_i = 1$. Using the same reasoning, we also have:

$$\begin{aligned}
\mathbb{E}[Y_i|R_i = 0] &= \mathbb{E}[Y_i^{1,0}D_i + Y_i^{0,0}(1 - D_i)|R_i = 0] \\
&= \mathbb{E}[Y_i^0|R_i = 0] \\
&= \mathbb{E}[Y_i^0].
\end{aligned}$$

A direct application of the formula for the Bloom estimator proves the result. □

3.3.1.2 Identification of ITE

The previous proof does not give a lot of intuition of how TT is identified in the Randomization After Eligibility design. In order to gain more insight, we are going to decompose the Bloom estimator, and have a look at its numerator. The numerator of the Bloom estimator is a With/Without comparison, and it identifies, under fairly light conditions, another causal effect, the Intention to Treat Effect (ITE).

Let me first define the ITE:
Definition 3.8 (Intention to Treat Effect)**.** In a Randomization After Eligibility design, the Intention to Treat Effect (ITE) is the effect of receiving information about eligiblity among eligibles:

$$\Delta^Y_{ITE} = \mathbb{E}[Y_i^{D_i^1,1} - Y_i^{D_i^0,0}|E_i = 1].$$

Receiving information about eligibility has two impacts, in the general framework that we have delineated so far: first, it triggers some individuals into the treatment (those for which $D_i^1 \neq 0$); second, it might have a direct effect on outcomes ($Y_i^{d,1} \neq Y_i^{d,0}$). This second effect is the effect of annoucing eligiblity that does not goes through participation into the program. For example, it is possible that announcing eligibility to a retirement program makes me save more for retirement, even if I end up not taking up the proposed program.

The two causal channels that are at work within the ITE can be seen more clearly after some manipulations:

$$\begin{aligned}
\Delta^Y_{ITE} &= \mathbb{E}[Y_i^{1,1}D_i^1 + Y_i^{0,1}(1-D_i^1) - (Y_i^{1,0}D_i^0 + Y_i^{0,0}(1-D_i^0))|E_i=1] \\
&= \mathbb{E}[Y_i^{1,1}D_i^1 + Y_i^{0,1}(1-D_i^1) - (Y_i^{0,0}(D_i^1+1-D_i^1))|E_i=1] \\
&= \mathbb{E}[(Y_i^{1,1}-Y_i^{0,0})D_i^1 + (Y_i^{0,1}-Y_i^{0,0})(1-D_i^1)|E_i=1] \\
&= \mathbb{E}[Y_i^{1,1}-Y_i^{0,0}|D_i^1=1,E_i=1]\Pr(D_i^1=1|E_i=1) \\
&\quad + \mathbb{E}[Y_i^{0,1}-Y_i^{0,0}|D_i^1=0,E_i=1]\Pr(D_i^1=0|E_i=1),
\end{aligned} \tag{3.1}$$

where the first equality follows from Assumption 3.6 and the second equality uses the fact that $D_i^0 = 0$, $\forall i$.

We can now see that the ITE is composed of two terms: the first term captures the effect of annoucing eligibility on those who decide to participate into the program; the second term captures the effect of announcing eligibility on those who do not participate into the program. Both of these effects are weighted by the respective proportions of those reacting to the eligibility annoucement by participating and by not participating respectively.

Now, in order to see how the ITE "contains" the TT, we can use the following theorem:
Theorem 3.4 (From ITE to TT)**.** *Under Assumptions 3.5, 3.6 and 3.7, ITE is equal to TT multiplied by the proportion of individuals taking up the treatment after eligibility has been announced:*

$$\Delta^Y_{ITE} = \Delta^Y_{TT}\Pr(D_i^1=1|E_i=1).$$

Proof. Under Assumption 3.7, Equation (3.1) becomes:

$$\begin{aligned}
\Delta^Y_{ITE} &= \mathbb{E}[Y_i^1 - Y_i^0|D_i^1=1,E_i=1]\Pr(D_i^1=1|E_i=1) \\
&\quad + \mathbb{E}[Y_i^0 - Y_i^0|D_i^1=0,E_i=1]\Pr(D_i^1=0|E_i=1) \\
&= \mathbb{E}[D_i^1(Y_i^1-Y_i^0)|R_i=1,E_i=1] \\
&= \mathbb{E}[Y_i^1 - Y_i^0|D_i^1=1,R_i=1,E_i=1]\Pr(D_i^1=1|R_i=1,E_i=1) \\
&= \mathbb{E}[Y_i^1 - Y_i^0|D_i=1,E_i=1]\Pr(D_i^1=1|E_i=1),
\end{aligned}$$

where the first equality follows from Assumption 3.7, the second from Bayes' rule and Assumptions 3.5, the third from Bayes' rule and the last from the fact that $D_i^1 = 1, R_i = 1 \Leftrightarrow D_i = 1$. □

The previous theorem shows that Assumption 3.7 shuts down any direct effect of the announcement of eligibility on outcomes. As a consequence of this assumption, the only impact that an eligibility annoucement has on outcomes is through participation into the program. Hence, the ITE is equal to TT multiplied by the proportion of people taking up the treatment when eligibility is announced.

In order to move from the link between TT and ITE to the mechanics of the Bloom estimator, we need two additional identification results. The first result shows that ITE can be identified under fairly light conditions by a WW estimator. The second result shows that the proportion of people taking up the treatment when eligiblity is announced is also easily estimated from the data.
Theorem 3.5 (Identification of ITE with Randomization After Eligibility)**.** *Under Assumptions 3.5 and 3.6, ITE is identified by the With/Without comparison among eligibles:*

$$\Delta^Y_{ITE} = \Delta^Y_{WW|E=1}.$$

Proof.

$$\begin{aligned}\Delta^Y_{WW|E=1} &= \mathbb{E}[Y_i|R_i=1,E_i=1]-\mathbb{E}[Y_i|R_i=0,E_i=1]\\ &= \mathbb{E}[Y_i^{D_i^1,1}|R_i=1,E_i=1]-\mathbb{E}[Y_i^{D_i^0,0}|R_i=0,E_i=1]\\ &= \mathbb{E}[Y_i^{D_i^1,1}|E_i=1]-\mathbb{E}[Y_i^{D_i^0,0}|E_i=1],\end{aligned}$$

where the second equality follows from Assumption 3.6 and the third from Assumption 3.5. □

Theorem 3.6 (Identification of $\Pr(D_i^1=1|E_i=1)$)**.** *Under Assumptions 3.5 and 3.6,* $\Pr(D_i^1=1|E_i=1)$ *is identified by the proportion of people taking up the offered treatment when informed about their eligibility status:*

$$\Pr(D_i^1=1|E_i=1)=\Pr(D_i=1|R_i=1,E_i=1).$$

Proof.

$$\begin{aligned}\Pr(D_i=1|R_i=1,E_i=1) &= \Pr(D_i^1=1|R_i=1,E_i=1)\\ &= \Pr(D_i^1=1|E_i=1),\end{aligned}$$

where the first equality follows from Assumption 3.6 and the second from Assumption 3.5. □

Corollary 3.1 (Bloom estimator and ITE)**.** *It follows from Theorems 3.5 and 3.6 that, under Assumptions 3.5 and 3.6, the Bloom estimator is equal to the ITE divided by the propotion of agents taking up the program when eligible:*

$$\Delta^Y_{Bloom|E=1}=\frac{\Delta^Y_{ITE}}{\Pr(D_i^1=1|E_i=1)}.$$

As a consequence of Corollary 3.1, we see that the Bloom estimator reweights the ITE, the effect of receiving information about eligibility, by the proportion of people reacting to the eligibility by participating in the program. From Theorem 3.4, we know that this ratio will be equal to TT if the Assumption 3.7 also holds, so that all the impact of the eligibility annoucement stems from entering the program. The eligibility annoucement serves as an instrument for program participation.

Remark. The design using Randomization After Eligibility seems like magic. You do not assign randomly the program, but information about the eligiblity status, but you can recover the effect of the program anyway. How does this magic work? Randomization After Eligibility is also less intrusive than Randomization After Self-Selection. With the latter design, you have to actively send away individuals that have expressed an interest for entering the program. This is harsh. With Randomization After Eligibility, you do not have to send away people expressing interest after being informed. And it seems that you are not paying a price for that, since you are able to recover the same TT parameter. Well, actually, you are going to pay a price in terms of larger sampling noise.

The intuition for all that can be delineated using the very same apparatus that we have developed so far. So here goes. Under the assumptions made so far, it is easy to show that (omitting the conditioning on $E_i=1$ for simplicity):

$$\begin{aligned}\Delta^Y_{WW|E=1} = {} & \mathbb{E}[Y_i^{1,1}|D_i^1=1,R_i=1]\Pr(D_i^1=1|R_i=1)\\ & -\mathbb{E}[Y_i^{0,0}|D_i^1=1,R_i=0]\Pr(D_i^1=1|R_i=0)\\ & +\mathbb{E}[Y_i^{0,1}|D_i^1=0,R_i=1]\Pr(D_i^1=0|R_i=1)\\ & -\mathbb{E}[Y_i^{0,0}|D_i^1=0,R_i=0]\Pr(D_i^1=0|R_i=0).\end{aligned}$$

The first part of the equation is due to the difference in outcomes between the two treatment arms for people that take up the program when eligibility is announced. The second part is due to the difference in outcomes between the two treatment arms for people that do not take up the program when eligibility is announced. This second part cancels out under Assumption 3.5 and 3.7.

But this cancelling out only happens in the population. In a given sample, the sample equivalents to the two members of the second part of the equation do not have to be equal, and thus they do not cancel out, generating additional sampling noise compared to the Randomization After Self-Selection design. Indeed, in the Randomization After Self-Selection design, you observe the population with $D_i^1 = 1$ in both the treatment and control arms (you actually observe this population before randomizing the treatment within it), and you can enforce that the effect on $D_i^1 = 0$ should be zero, under your assumptions. In the Randomization After Eligiblity design, you do not observe the population with $D_i^1 = 1$ in the control arm, and you cannot enforce the equality of the outcomes for those with $D_i^1 = 0$ present in both arms. You have to rely on the sampling estimates to make this cancellation, and that generates sampling noise.
Remark. In practice, we use a pseudo-RNG to allocate the randomized annoucement of the eligibility status:

$$\begin{aligned} R_i^* &\sim \mathcal{U}[0,1] \\ R_i &= \begin{cases} 1 & \text{if } R_i^* \leq .5 \wedge E_i = 1 \\ 0 & \text{if } R_i^* > .5 \wedge E_i = 1 \end{cases} \\ D_i &= \mathbb{1}[\bar{\alpha} + \theta\bar{\mu} - C_i \geq 0 \wedge E_i = 1 \wedge R_i = 1] \end{aligned}$$

Example 3.14. In our numerical example, we can actually use the same sample as we did for Randomization After Self-Selection. I have to generate it again, though, since I am going to allocate R_i differently.

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
YB <- exp(yB)
E <- ifelse(YB<=param["barY"],1,0)
V <- rnorm(N,0,param["sigma2V"])
Dindex <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
Dstar <- ifelse(Dindex>=0 & E==1,1,0)
epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
U0 <- param["rho"]*UB + epsilon
y0 <- mu +  U0 + param["delta"]
alpha <- param["baralpha"]+  param["theta"]*mu + eta
y1 <- y0+alpha
Y0 <- exp(y0)
Y1 <- exp(y1)
```

The value of TT in our example is the same as the one in the Randomization After Self-Selection case. TT in the population is equal to 0.17.

Let's now compute the value of ITE in the population. In our model, exclusion restriction holds, so that we can use the fact that $ITE = TT \Pr(D_i^1 = 1|E_i = 1)$. We thus only need to compute $\Pr(D_i^1 = 1|E_i = 1)$:

$$\Pr(D_i^1 = 1|E_i = 1) = \Pr(D_i^* \geq 0|y_i^B \leq \bar{y}).$$

I can again use the package `tmvtnorm` to compute that probability. It is indeed equal to $1-\Pr(D_i^* < 0|y_i^B \leq \bar{y})$, where $\Pr(D_i^* < 0|y_i^B \leq \bar{y})$ is the cumulative density of D_i^* conditional on $y_i^B \leq \bar{y}$, *i.e.* the marginal cumulative

of the third variable of the truncated trivariate normal (μ_i, y_i^B, D_i^*) where the first variable is not truncated and the second one is truncated at $\bar{y}$.

```
lower.cut <- c(-Inf,-Inf,-Inf)
upper.cut <- c(Inf,log(param['barY']),Inf)
prD1.elig <- 1-ptmvnorm.marginal(xn=0,n=3,mean=mean.mu.yB.Dstar,sigma=cov.mu.yB.Dstar,lower=lower.cut,u
delta.y.ite <- delta.y.tt*prD1.elig
```

$\Pr(D_i^1 = 1|E_i = 1) = 0.459$. As a consequence, ITE in the population is equal to 0.17 * 0.459 $\approx$ 0.078. In the sample, the value of ITE and TT are equal to:

```
delta.y.tt.sample <- mean(y1[E==1 & Dstar==1]-y0[E==1 & Dstar==1])
delta.y.ite.sample <- delta.y.tt.sample*mean(Dstar[E==1])
```

$\Delta^y_{ITE_s} = 0.068$ and $\Delta^y_{TT_s} = 0.187$.

Now, we can allocate the randomized treatment and let potential outcomes be realized:

```
#random allocation among eligibles
Rs <- runif(N)
R <- ifelse(Rs<=.5 & E==1,1,0)
Ds <- ifelse(Dindex>=0 & E==1 & R==1,1,0)
y <- y1*Ds+y0*(1-Ds)
Y <- Y1*Ds+Y0*(1-Ds)
```

3.3.2 Estimating the ITE and the TT

In general, we start the analysis of Randomization After Eligibility by estimating the ITE. Then, we provide the TT by dividing the ITE by the proportion of participants among the eligibles.

Actually, this procedure is akin to an instrumental variables estimator and we will see that the Bloom estimator is actually an IV estimator. The ITE estimation step corresponds to the reduced form in a classical IV approach. Estimation of the proportion of participants is the first stage in a IV approach. Estimation of the TT corresponds to the structural equation step of an IV procedure.

3.3.2.1 Estimating the ITE

Estimation of the ITE relies on the WW estimator, in general implemented using OLS. It is similar to the estimation of ATE and TT in the Brute Force and Randomization After Self-Selection designs.

3.3.2.1.1 Using the WW estimator

Estimation of the ITE can be based on the WW estimator among eligibles.

$$\hat{\Delta}^Y_{WW|E=1} = \frac{1}{\sum_{i=1}^N E_i R_i} \sum_{i=1}^N Y_i E_i R_i - \frac{1}{\sum_{i=1}^N E_i (1 - R_i)} \sum_{i=1}^N E_i Y_i (1 - R_i).$$

Example 3.15. In our numerical example, we can form the WW estimator among eligibles:

```
delta.y.ww.elig <- mean(y[R==1 & E==1])-mean(y[R==0 & E==1])
```

WW among eligibles is equal to 0.069.

3.3.2.1.2 Using OLS

As we have already seen before, the WW estimator is equivalent to OLS with one constant and no control variables. As a consequence, we can estimate the ITE using the OLS estimate of β in the following regression run on the sample with $E_i = 1$:

$$Y_i = \alpha + \beta R_i + U_i.$$

By construction, $\hat{\beta}_{OLSR|E=1} = \hat{\Delta}^Y_{WW|E=1}$.
Example 3.16. In our numerical example, we can form the WW estimator among eligibles:

```
reg.y.ols.elig <- lm(y[E==1]~R[E==1])
delta.y.ols.elig <- reg.y.ols.elig$coef[2]
```

$\hat{\beta}_{OLSR|E=1}$ is equal to 0.069. Remember that ITE in the population is equal to 0.078.

3.3.2.1.3 Using OLS conditioning on covariates

Again, as in the previous designs, we can compute ITE by using OLS conditional on covariates. Parametrically, we can run the following OLS regression among eligibles (with $E_i = 1$):

$$Y_i = \alpha + \beta R_i + \gamma' X_i + U_i.$$

The OLS estimate of β estimates the ITE.

Again: Needed: proof. Especially check whether we need to center covariates at the mean of the treatment group. I think so.

We can also use Matching to obtain a nonparametric estimator.
Example 3.17. Let us compute the OLS estimator conditioning on y_i^B:

```
reg.y.R.yB.ols.elig <- lm(y[E==1] ~ R[E==1] + yB[E==1])
```

Our estimate of ITE after conditioning on y_i^B is 0.065. I do not have time to run the simulations, but it is highly likely that the sampling noise is lower after conditioning on y_i^B.

I do not have time to run the simulations, but it is highly likely that the sampling noise is lower after conditioning on y_i^B.

3.3.2.2 Estimating TT

We can estimate TT either using the Bloom estimator, or using the IV estimator, which is equivalent to a Bloom estimator in the Eligibility design.

3.3.2.2.1 Using the Bloom estimator

Using the Bloom estimator, we simply compute the numerator of the Bloom estimator and divide it by the estimated proportion of eligible individuals with $R_i = 1$ that have chosen to take the program.

$$\hat{\Delta}^Y_{WW|D=1} = \frac{\frac{1}{\sum_{i=1}^N E_i R_i} \sum_{i=1}^N Y_i E_i R_i - \frac{1}{\sum_{i=1}^N E_i(1-R_i)} \sum_{i=1}^N E_i Y_i (1-R_i)}{\frac{1}{\sum_{i=1}^N E_i R_i} \sum_{i=1}^N D_i E_i R_i}.$$

Example 3.18. Let's see how the Boom estimator works in our example.

The numerator of the Bloom estimator is the ITE that we have just computed: 0.069. The denominator of the Bloom estimator is equal to the proportion of eligible individuals with $R_i = 1$ that have chosen to take the program: 0.342.

```
delta.y.R.bloom.elig <- (mean(y[R==1 & E==1])-mean(y[R==0 & E==1]))/mean(Ds[R==1 & E==1])
```

The resulting estimate of TT is 0.203. It is rather far from the population or sample estimates: 0.17 and 0.187 respectively. What happened? The error seems to come from noise in the denominator of the Bloom estimator. In the ITE estimation, the true ITEs in the population and sample are 0.078 and 0.068 respectively and our estimate is equal to 0.069, so that's fine. In the denominator, the proportion of randomized eligibles that take the program is equal to 0.342 while the true proportions in the population and in the sample are 0.459 and 0.364 respectively. So we do not have enough invited eligibles getting into the program, and the ones who do have unusually large outcomes. These two sampling errors combine to blow up the estimate of TT.

3.3.2.2.2 Using IV

There is a very useful results, similar to the one stating that the WW estimator is equivalent to an OLS estimator: in the Eligiblity design, the Bloom estimator is equivalent to an IV estimator:

Theorem 3.7 (Bloom is IV)**.** *Under the assumption that there is at least one individual with $R_i = 1$ and one individual with $D_i = 1$, the coefficient β in the following regression estimated among eligibles using R_i as an IV*

$$Y_i = \alpha + \beta D_i + U_i$$

is the Bloom estimator in the Eligibility Design:

$$\begin{aligned}\hat{\beta}_{IV} &= \frac{\frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i\left(Y_i - \frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i Y_i\right)\left(R_i - \frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i R_i\right)}{\frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i\left(D_i - \frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i D_i\right)\left(R_i - \frac{1}{\sum_{i=1}^N E_i}\sum_{i=1}^N E_i R_i\right)} \\ &= \frac{\frac{1}{\sum_{i=1}^N E_i R_i}\sum_{i=1}^N Y_i R_i E_i - \frac{1}{\sum_{i=1}^N (1-R_i)E_i}\sum_{i=1}^N Y_i(1-R_i)E_i}{\frac{1}{\sum_{i=1}^N E_i R_i}\sum_{i=1}^N D_i R_i E_i}.\end{aligned}$$

Proof. The proof is straightforward using Lemma 3.1 below and setting $D_i = 0$ when $R_i = 0$. □

Example 3.19. In our numerical example, we have:

```
reg.y.R.2sls.elig <- ivreg(y[E==1]~Ds[E==1]|R[E==1])
```

$\hat{\beta}_{IV} = 0.203$ which is indeed equal to the Bloom estimator ($\hat{\Delta}^y_{Bloom} = 0.203$).

3.3.2.2.3 Using IV conditional on covariates

We can improve on the precision of our 2SLS estimator by conditioning on observed covariates. Parametrically estimating the following equation with R_i and X_i as instruments on the sample with $E_i = 1$:

$$Y_i = \alpha + \beta D_i + \gamma' X_i + U_i.$$

Proof? Do we need to center covariates to their mean in the treatment group?

Nonparametric estimation using Frolich's Wald matching estimator.}
Example 3.20. In our numerical example, we have:

```
reg.y.R.yB.2sls.elig <- ivreg(y[E==1] ~ Ds[E==1] + yB[E==1] | R[E==1] + yB[E==1])
```

As a consequence, $\hat{\Delta}^{y}_{Bloom(X)} = 0.191$.

Does conditioning on covariates improve precision? Let's run some Monte-Carlo somulations in order to check for that.

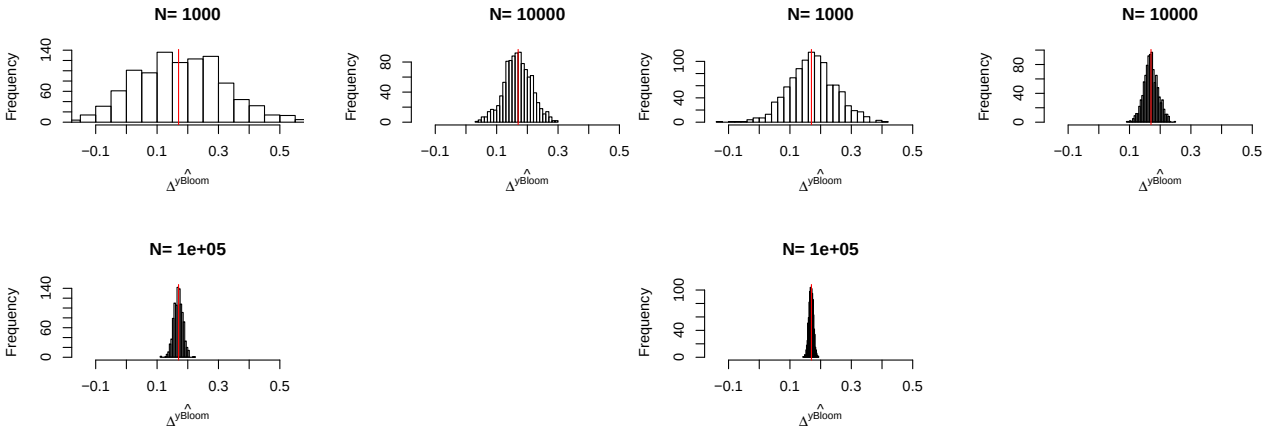
```
monte.carlo.elig <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  E <- ifelse(YB<=param["barY"],1,0)
  V <- rnorm(N,0,param["sigma2V"])
  Dindex <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
  Dstar <- ifelse(Dindex>=0 & E==1,1,0)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)

  #random allocation among self-selected
  Rs <- runif(N)
  R <- ifelse(Rs<=.5 & E==1,1,0)
  Ds <- ifelse(Dindex>=0 & E==1 & R==1,1,0)
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  reg.y.R.2sls.elig <- ivreg(y[E==1]~Ds[E==1]|R[E==1])
  return(reg.y.R.2sls.elig$coef[2])
}

simuls.elig.N <- function(N,Nsim,param){
  simuls.elig <- matrix(unlist(lapply(1:Nsim,monte.carlo.elig,N=N,param=param)),nrow=Nsim,ncol=1,byrow=T
  colnames(simuls.elig) <- c('Bloom')
  return(simuls.elig)
}

sf.simuls.elig.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=8)
  sfLibrary(AER)
  sim <- matrix(unlist(sfLapply(1:Nsim,monte.carlo.elig,N=N,param=param)),nrow=Nsim,ncol=1,byrow=TRUE)
  sfStop()
  colnames(sim) <- c('Bloom')
  return(sim)
}
```

```
Nsim <- 1000
#Nsim <- 10
#N.sample <- c(100,1000,10000,100000)
N.sample <- c(1000,10000,100000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.elig <- lapply(N.sample,sf.simuls.elig.N,Nsim=Nsim,param=param)
names(simuls.elig) <- N.sample
```

```
monte.carlo.elig.yB <- function(s,N,param){
  set.seed(s)
  mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
  UB <- rnorm(N,0,sqrt(param["sigma2U"]))
  yB <- mu + UB
  YB <- exp(yB)
  E <- ifelse(YB<=param["barY"],1,0)
  V <- rnorm(N,0,param["sigma2V"])
  Dindex <- param["baralpha"]+param["theta"]*param["barmu"]-param["barc"]-param["gamma"]*mu-V
  Dstar <- ifelse(Dindex>=0 & E==1,1,0)
  epsilon <- rnorm(N,0,sqrt(param["sigma2epsilon"]))
  eta<- rnorm(N,0,sqrt(param["sigma2eta"]))
  U0 <- param["rho"]*UB + epsilon
  y0 <- mu +  U0 + param["delta"]
  alpha <- param["baralpha"]+  param["theta"]*mu + eta
  y1 <- y0+alpha
  Y0 <- exp(y0)
  Y1 <- exp(y1)

  #random allocation among self-selected
  Rs <- runif(N)
  R <- ifelse(Rs<=.5 & E==1,1,0)
  Ds <- ifelse(Dindex>=0 & E==1 & R==1,1,0)
  y <- y1*Ds+y0*(1-Ds)
  Y <- Y1*Ds+Y0*(1-Ds)
  reg.y.R.yB.2sls.elig <- ivreg(y[E==1] ~ Ds[E==1] + yB[E==1] | R[E==1] + yB[E==1])
  return(reg.y.R.yB.2sls.elig$coef[2])
}

simuls.elig.yB.N <- function(N,Nsim,param){
  simuls.elig.yB <- matrix(unlist(lapply(1:Nsim,monte.carlo.elig.yB,N=N,param=param)),nrow=Nsim,ncol=1,
  colnames(simuls.elig.yB) <- c('Bloom')
  return(simuls.elig.yB)
}

sf.simuls.elig.yB.N <- function(N,Nsim,param){
  sfInit(parallel=TRUE,cpus=8)
  sfLibrary(AER)
  sim <- matrix(unlist(sfLapply(1:Nsim,monte.carlo.elig.yB,N=N,param=param)),nrow=Nsim,ncol=1,byrow=TRU
  sfStop()
  colnames(sim) <- c('Bloom')
  return(sim)
}
```

```
Nsim <- 1000
#Nsim <- 10
#N.sample <- c(100,1000,10000,100000)
N.sample <- c(1000,10000,100000)
#N.sample <- c(100,1000)
#N.sample <- c(100)

simuls.elig.yB <- lapply(N.sample,sf.simuls.elig.yB.N,Nsim=Nsim,param=param)
names(simuls.elig.yB) <- N.sample
```

```
par(mfrow=c(2,2))
for (i in 1:length(simuls.elig)){
  hist(simuls.elig[[i]][,'Bloom'],breaks=30,main=paste('N=',as.character(N.sample[i])),xlab=expression(
  abline(v=delta.y.tt,col="red")
}
par(mfrow=c(2,2))
for (i in 1:length(simuls.elig.yB)){
  hist(simuls.elig.yB[[i]][,'Bloom'],breaks=30,main=paste('N=',as.character(N.sample[i])),xlab=expressi
  abline(v=delta.y.tt,col="red")
}
```



(a) *Bloom* (b) *Bloom*(X)

Figure 3.6: Distribution of the *Bloom* and *Bloom*(X) estimators with randomization after eligibility over replications of samples of different sizes

We can take three things from Figure 3.6:

1. Problems with the IV estimator appear with $N = 100$ (probably because there are some samples where no one is treated).
2. Sampling noise from randomization after eligibility is indeed larger than sampling noise from randomization after self-selection.
3. Conditioning on covariates helps.

3.3.3 Estimating sampling noise

As always, we can estimate samling noise either using the CLT or resampling methods. Using the CLT, we can derive the following formula for the distribution of the Bloom estimator:

Theorem 3.8 (Asymptotic Distribution of $\hat{\Delta}^Y_{Bloom}$)**.** *Under Assumptions 3.5, 3.6 and 3.7 and assuming that*

there is at least one individual with $R_i = 1$ and one individual with $D_i = 1$, we have (keeping the conditioning on $E_i = 1$ implicit):

$$\sqrt{N}(\hat{\Delta}^Y_{Bloom} - \Delta^Y_{TT}) \xrightarrow{d} \mathcal{N}\left(0, \frac{1}{(p^{DR})^2}\left[\left(\frac{p^D}{p^R}\right)^2 \frac{\mathbb{V}[Y_i|R_i=0]}{1-p^R} + \left(\frac{1-p^D}{1-p^R}\right)^2 \frac{\mathbb{V}[Y_i|R_i=1]}{p^R}\right]\right),$$

with $p^D = \Pr(D_i = 1)$, $p^R = \Pr(R_i = 1)$ and $(p^{DR} = \Pr(D_i = 1|R_i = 1)$.

Proof. The proof is immediate using Lemma **??** in the appendix and setting $p^{AT} = 0$. □

Remark. Theorem 3.8 shows that there is a price to pay for not randomizing after self-selection. This price is a decrease in precision. The variance of the estimator is weighted by $\frac{1}{(\Pr(D_i=1|R_i=1))^2}$. This means that the effective sample size is equal to the number of individuals that take up the treatment when offered. We generaly call these individuals "compliers," since they comply with the treatment assignment. Sampling noise is of the same order of magnitude as the number of compliers You might have very low precision despite a very large sample size if you have a very small proportion of compliers.
Remark. In order to compute an estimate of the sampling noise of hte Bloom estimator, we can either use the plug-in formula from Theorem 3.8 or use the IV standard errors robust to heteroskedasticity. Here is a simple function in order to compute the plug-in estimator:

```
var.RAE.plugin <- function(pDR,pD,pR,V0,V1,N){
  return(((pD/pR)^2*(V0/(1-pR))+((1-pD)/(1-pR))^2*(V1/pR))/(N*pDR^2))
}
```

Example 3.21. Let us derive the CLT-based estimates of sampling noise using both the plug-in estimator and the IV standard errors without conditioning on covariates first. For the sake of the example, I'm working with a sample of size $N = 1000$.

```
sn.RAE.simuls <- 2*quantile(abs(simuls.elig[['1000']][,'Bloom']-delta.y.tt),probs=c(0.99))
sn.RAE.IV.plugin <- 2*qnorm((.99+1)/2)*sqrt(var.RAE.plugin(pDR=mean(Ds[E==1 & R==1]),pD=mean(Ds[E==1]),
sn.RAE.IV.homo <- 2*qnorm((.99+1)/2)*sqrt(vcov(reg.y.R.2sls.elig)[2,2])
sn.RAE.IV.hetero <- 2*qnorm((.99+1)/2)*sqrt(vcovHC(reg.y.R.2sls.elig,type='HC2')[2,2])
```

True 99% sampling noise (from the simulations) is 0.757. 99% sampling noise estimated using the plug-in estimator is 0.921. 99% sampling noise estimated using default IV standard errors is 1.069. 99% sampling noise estimated using heteroskedasticity robust IV standard errors is 0.92.

Conditioning on covariates:

```
sn.RAE.simuls.yB <- 2*quantile(abs(simuls.elig.yB[['1000']][,'Bloom']-delta.y.tt),probs=c(0.99))
sn.RAE.IV.homo.yB <- 2*qnorm((.99+1)/2)*sqrt(vcov(reg.y.R.yB.2sls.elig)[2,2])
sn.RAE.IV.hetero.yB <- 2*qnorm((.99+1)/2)*sqrt(vcovHC(reg.y.R.yB.2sls.elig,type='HC2')[2,2])
```

True 99% sampling noise (from the simulations) is 0.393. 99% sampling noise estimated using default IV standard errors is 0.457. 99% sampling noise estimated using heteroskedasticity robust IV standard errors is 0.454.
Remark. Sampling noise in the Randomization After Eligibility design seems larger than sampling noise in the Randomization After Self-Selection design.

In the Randomization After Self-Selection design, sampling noise with $N = 1000$ is equal to 0.55. In the Randomization After Eligibility design, sampling noise with $N = 1000$ is equal to 0.76. Why such a difference? Both designs have the same effective sample size.

In the Randomization After Self-Selection design, the effective sample size N_e^{RASS} is the number of eligible individuals that apply to take up the program: $N_e^{RASS} = N \Pr(D_i = 1|E_i = 1) \Pr(E_i = 1)$. In our example, $N_e^{RASS} = 1000 * 0.459 * 0.218 = 100$.

In the Randomization After Eligibility design, the sample size on which the regressions are performed is N^{RAE}, the number of eligible individuals: $N^{RAE} = N \Pr(E_i = 1)$. In our example, $N^{RAE} = 1000 * 0.459 = 459$. But the effective sample size for the Randomization After Eligibility design is actually equal to the one in the Randomization After Self-Selection design because only compliers matter for the precision of the Bloom estimator, as Theorem 3.8 shows. Thus $N_e^{RASS} = N_e^{RAE}$.

Why then is sampling noise much larger in the Randomization After Eligibility design? Probably because the Bloom estimator cannot enforce the fact that the impact of the program on non compliers is zero. It has to estimate the average outcome of non compliers in both treatment arms and hope that they cancel. In real samples, they won't, increasing the size of sampling noise.

3.4 Encouragement Design

In an Encouragement Design, we randomly select two groups among the eligibles, as in Randomization After Eligibility. Treated individuals randomly receive an encouragement to participate in the program and decide whether they want to comply with the encouragement and join the program. Individuals in the control group do not receive an encouragement, but they can still decide to self-select in the program. An Encouragement Design differs from Randomization After Eligibility mainly by not barring entry into the programs to individuals in the control group. If successful, the encouragement generates a higher level of take up of the program in the treatment group than in the control group.

In an Encouragement Design, we can recover the causal effect of the treatment not on all the treated but on those treated whose participation into the program has been triggered by the enocuragement. These individuals reacting to the encouragement by participating in the program are usually called compliers. The effect of the treatment on the compliers is called the Local Average Treatment Effect.

The main identification for Encouragement designs is that a Wald ratio (an IV estimator) recovers the LATE. It is due to Imbens and Angrist (1994). Let's detail this result.

3.4.1 Identification

Write the correct assumptions

Monotonicity is missing.

Here are the two assumptions that we are going to need in order to identify the LATE on top of Independence Among Eligibles:
Definition 3.9 (Encouragement Design Validity)**.** We assume that no eligibles that has been randomized out can take the treatment and that the randomized allocation of the program does not interfere with how potential outcomes and self-selection are generated:

$$D_i = D_i^1 R_i + (1 - R_i) D_i^0$$
$$Y_i = \begin{cases} Y_i^{1,1} & \text{if } (R_i = 1 \text{ and } D_i = 1) \\ Y_i^{0,1} & \text{if } (R_i = 1 \text{ and } D_i = 0) \\ Y_i^{0,0} & \text{if } R_i = 0 \end{cases}$$

with $Y_i^{1,1}$, $Y_i^{0,1}$, $Y_i^{0,0}$, D_i^1 and D_i^0 the same potential outcomes and self-selection decisions as in a routine allocation of the treatment.

We need another assumption:

Definition 3.10 (Exclusion Restriction of Encouragement)**.** We assume that there is no direct effect of being informed about eligibliity to the program on outcomes:

$$Y_i^{1,1} = Y_i^{1,0} = Y_i^1$$
$$Y_i^{0,1} = Y_i^{0,0} = Y_i^0.$$

Under these assumptions, we have the following result:
Theorem 3.9 (Identification of TT With Randomization After Eligibility)**.** *Under Independence Among Eligibles, Randomization After Eligibility Validity and Exclusion Restriction of Eligibility, the Wald estimator among eligibles identifies LATE:*

$$\Delta^Y_{Wald|E=1} = \Delta^Y_{LATE}$$

with:

$$\Delta^Y_{Wald|E=1} = \frac{\Delta^Y_{WW|E=1}}{\Pr(D_i = 1|R_i = 1, E_i = 1) - \Pr(D_i = 1|R_i = 0, E_i = 1)}\Delta^Y_{LATE} \quad = \mathbb{E}[Y_i^1 - Y_i^0|D_i^1 = 1, D_i^0 = 0]$$

3.4.2 Estimating sampling noise

For simplicity, in all that follows, I am setting $E_i = 1$ for everyone, so that N is the number of eligible individuals.
Lemma 3.1 (Wald is IV)**.** *Under the assumption that there is at least one individual with $R_i = 1$ and one individual with $D_i = 1$, the coefficient β in the following regression estimated using R_i as an IV:*

$$Y_i = \alpha + \beta D_i + U_i$$

is the Wald estimator in the Encouragement Design and the Bloom estimator in the Eligibility Design:

$$\begin{aligned}\hat{\beta}_{IV} &= \frac{\frac{1}{N}\sum_{i=1}^N \left(Y_i - \frac{1}{N}\sum_{i=1}^N Y_i\right)\left(R_i - \frac{1}{N}\sum_{i=1}^N R_i\right)}{\frac{1}{N}\sum_{i=1}^N \left(D_i - \frac{1}{N}\sum_{i=1}^N D_i\right)\left(R_i - \frac{1}{N}\sum_{i=1}^N R_i\right)} \\ &= \frac{\frac{1}{\sum_{i=1}^N R_i}\sum_{i=1}^N Y_i R_i - \frac{1}{\sum_{i=1}^N (1-R_i)}\sum_{i=1}^N Y_i(1-R_i)}{\frac{1}{\sum_{i=1}^N R_i}\sum_{i=1}^N D_i R_i - \frac{1}{\sum_{i=1}^N (1-R_i)}\sum_{i=1}^N D_i(1-R_i)}.\end{aligned}$$

Proof. See in section A.2.1 in the appendix. □

3.5 Threats to the validity of RCTs

Chapter 4

Natural Experiments

Chapter 5

Observational Methods

Part III

Additional Topics

Chapter 6

Power Analysis

Chapter 7

Placebo Tests

Chapter 8

Clustering

Chapter 9

LaLonde Tests

Chapter 10

Diffusion effects

Chapter 11

Distributional effects

Chapter 12

Meta-analysis and Publication Bias

When several research teams work on a similar topic, they obtain and publish several estimates for the same program of for similar programs. For example, teams of doctors regularly test the same treatment on different samples or populations in order to refine the estimated effect. Similarly, economists report on the effects of similar types of programs (Conditional and Unconditional Cash Transfers, Job Training Programs, microcredit, etc) implemented in different countries.

Meta-analysis aims at summarizing and synthetizing the available evidence with two main goals in mind:

1. Increasing precision by providing an average estimated effect combining several estimates
2. Explaining variations in treatment effectiveness by relating changes in effect size to changes in sample characteristics.

One key issue that meta-analysis has to face – actually, we all have to face it, meta-analysis simply makes it more apparent – is that of publication bias. Publication bias is due to the fact that referees and editors have a marked preference for publishing statistically significant results. The problem with this approach is that the distribution of published results is going to be censored on the left: we will have more statistically significant results in the published record, and as a consequence, the average published result will be an upward biased estimate of the true treatment effect in the population. This is potentially a very severe problem if the amount of censoring due to publication bias is large. Eventually, this hinges on the true distribution of treatment effects: if it is centered on zero or close to zero, we run the risk of having very large publication bias.

In this chapter, I present first the tools for meta-analysis, and I then move on to testing and correcting for publication bias.

12.1 Meta-analysis

There are several approaches and refinements to meta-analysis. In this section, I am going to present only the most important ones. I'll defer the reader to other more specialized publications if needed.

I first present the basics of meta-analysis: the constitution and structure of the sample. Second, I present the simplest method to aggregate effects: the weighted average. Third, I present tests for deciding whether the effects are from a homogeneous or a heterogeneous population. Fourth, I explain how to conduct a meta-analysis when effects are heterogeneous. Fifth, I cover the meta-regression that we use to account for heterogeneity in treatment effects. Finally, I explain why usual intuitive methods such as vote-counting are biased.

12.1.1 Basic setting

The basic setting for a meta-analysis is that you have access to a list of estimates for the effect of a given program and for their precision. These estimates come from the literature, searching published and unpublished sources alike. This data is usually collected after an extensive search of bibliographic databases. Then, one has to select among all the studies selected by the search the ones that are actualy relevant. This is the most excruciating part of a meta-analysis, since a lot of the studies selected by hte search algorithm are actually irrelevant. Finally, one has to extract from each relevant paper an estimate of the effect of the treatment and of its precision. In general, one tries to choose standardized estimates such as the effect size (see Section 2.1.6 for a definition) and its standard error. After all this process, we should end up with a dataset like: $\left\{(\hat{\theta}_k, \hat{\sigma}_k)\right\}_{k=1}^{N}$, with $\hat{\theta}_k$ the estimated effect size, $\hat{\sigma}_k$ its estimated standard error, and N the number of included studies.

Example 12.1. Let's see how such a dataset would look like? Let's build one from our simulations.

```
N.sample <- c(100,1000,10000,100000)
N.plot.ES.CLT <- c(10,7,2,1)
data.meta <- data.frame(ES=numeric(),
                        se=numeric())

se.ww.CLT.ES <- function(N,v1,v0,p){
  return(sqrt((v1/p+v0/(1-p))/N)/v0)
}

for (k in 1:length(N.sample)){
  set.seed(1234)
  simuls.ww[[k]]$se.ES <- se.ww.CLT.ES(N.sample[[k]],simuls.ww[[k]][,'V1'],simuls.ww[[k]][,'V0'],simuls
  test.ES <- simuls.ww[[k]][sample(N.plot.ES.CLT[[k]]),c('ES','se.ES')]
  test.ES$N <- rep(N.sample[[k]],N.plot.ES.CLT[[k]])
  data.meta <- rbind(data.meta,test.ES)
  }

data.meta$id <- 1:nrow(data.meta)
#data.meta$N <- factor(data.meta$N,levels(N.sample))

  ggplot(data.meta, aes(x=as.factor(id), y=ES)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=ES-qnorm((delta.2+1)/2)*se.ES, ymax=ES+qnorm((delta.2+1)/2)*se.ES), width=
      geom_hline(aes(yintercept=ES(param)), colour="#990000", linetype="dashed")+
      xlab("Studies")+
      ylab("Effect size")+
      theme_bw()
```

Figure 12.1 shows the resulting sample. I've selected 10 studies with $N = 100$, 7 studies with $N = 1000$, 2 studies with $N = 10\hat{}\{4\}$, and 1 study with $N = 10\hat{}\{5\}$. The studies are represented in that order, mimicking the increasing sample size of studies that accumulate evidence on a treatment, probably with studies with a small sample size at first, and only large studies at the end for the most promising treatments.

12.1.2 Meta-analysis as a weighted average

The key idea of meta-analysis is to combine the effect size estimates stemming from different studies, weighing them by their relative precision.

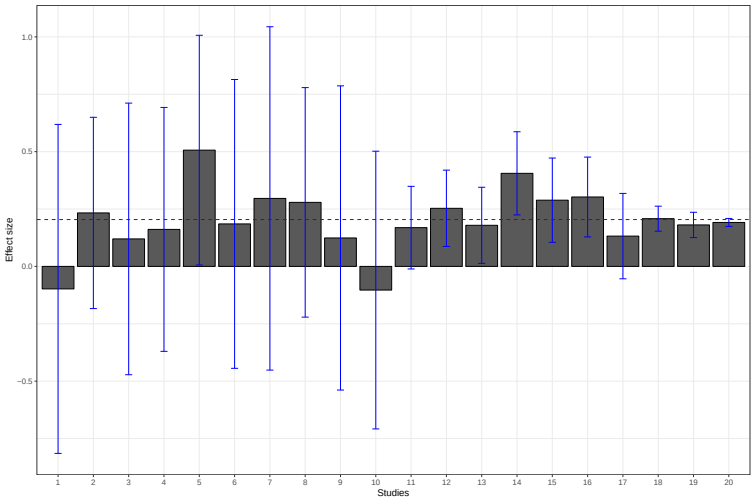


Figure 12.1: Example data set: effect sizes and confidence intervals with $\delta = 0.95$

Definition 12.1 (Weighted Meta-Analytic Estimator)**.** The weighted meta-analytic estimator is

$$\bar{\theta} = \sum_{k=1}^{N} w_k \hat{\theta}_k \text{ with } w_k = \frac{\frac{1}{\hat{\sigma}_k^2}}{\sum_{k=1}^{N} \frac{1}{\hat{\sigma}_k^2}}.$$

Under some assumptions, the estimator $\bar{\theta}$ converges to the true effect of the treatment. Let's delineate these assumptions.

Definition 12.2 (Homogeneous Treatment Effect)**.** Each $\hat{\theta}_k$ converges to the same treatment effect θ.

Assumption 12.2 imposes that all the studies have been drawn from the same population, where the treatment effect is a constant.

Definition 12.3 (Independence of Estimates)**.** The $\hat{\theta}_k$ are independent from each other.

Assumption 12.3 imposes that all the studies estimates are independent from each other. That means that they do not share sampling units and that they are not affected by common shocks.

Under these assumptions, we can show two important results.

Theorem 12.1 (Consistency of the Weighted Meta-Analytic Estimator)**.** *Under Assumptions 12.2 and 12.3, when the sample size of each study goes to infinity,* $\bar{\theta} \approx \theta$.

Proof. The Law of Large Number applied to each sample gives the fact that the estimator is a weighted sum of θ with weights summing to one. Hence the result. □

Theorem 12.1 says that the error we are making around the true effect of the treatment goes to zero as the sample size in each study decrease. This is great: aggregating the studies is thus going to get us to the truth.

Remark. One interesting question is whether Theorem 12.1 also holds when the size of the individual studies remains fixed and the number of studies goes to infinity, which seems a more natural way to do asymptotics in a meta-analysis. I'm pretty sure that is the case. Indeed, the studies constitute an enormous sample in which we take the average outcomes of the treated on the one hand and of the untreated on the other. These averages differ from the usual ones in the Law of Large Numbers only by the fact that the weights are not equal to one. But they (i) are independent from the outcomes and (ii) sum to one. As a consequence, I'm pretty sure the Law of Large Numbers also apply in this dimension.

Check if this is a consequence of Kolmogorov's Law of Large Numbers.

Theorem 12.2 (Asymptotic Distribution of the Weighted Meta-Analytic Estimator)**.** *Under Assumptions 12.2 and 12.3, when the sample size of each study goes to infinity,* $\bar{\theta} \xrightarrow{d} \mathcal{N}(\theta, \sigma^2)$*, with*

$$\sigma^2 = \frac{1}{\sum_{k=1}^{N} \frac{1}{\sigma_k^2}}.$$

Proof. **To do using the Lindenberg-Levy version of the Central Limit Theorem.** □

Theorem 12.2 shows that the distribution of the weighted meta-analytic estimator converges to a normal, which is very convenient in order to compute sampling noise. In order to obtain an estimator $\hat{\sigma}^2$ of the variance of the meta-analytic estimator, we can simply replace the individual variance terms by their estimates: $\hat{\sigma}_k^2$.
Remark. I've taken Theorem 12.2 from Hedges and Olkin, but I think it is much more interesting and correct when the asymptotics goes in the number of studies.
Remark. According to Hedges and Olkin, the weighted meta-analytic estimator is the most efficient estimator available.

```
wmae <- function(theta,sigma2){
  return(c(weighted.mean(theta,(1/sigma2)/(sum(1/sigma2))),1/sum(1/sigma2)))
}
```

Example 12.2. Let's use our meta-analytic estimator to estimate the effect size of our treatment.

The estimated treatment effect size with our sample is 0.19 ± 0.02. A very simple way to implement such an estimator in R is to use the `rma` command of the `metafor` package.

```
data.meta$var.ES <- data.meta$se.ES^2
meta.example.FE <- rma(yi = data.meta$ES,vi=data.meta$var.ES,method="FE")
summary(meta.example.FE)
```

```
##
## Fixed-Effects Model (k = 20)
##
##   logLik  deviance       AIC       BIC      AICc
##  16.1375   12.7060  -30.2751  -29.2793  -30.0529
##
## Test for Heterogeneity:
## Q(df = 19) = 12.7060, p-val = 0.8533
##
## Model Results:
##
## estimate      se     zval    pval   ci.lb   ci.ub
##   0.1950  0.0079  24.6975  <.0001  0.1795  0.2104  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As seen above, the `metafor` package yields a meta-analytic estimate of 0.19 ± 0.02, as we have found using the weighted meta-analytic estimator.

It is customary to present the results of a meta-analysis using a forest plot. A forest plows all the individual estimates along with the aggregated estimate. Figure 12.2 presents the forest plot for our example.

12.1.3 Constantly updated meta-analysis

Constantly updated meta-analysis performs the meta-analysis in a progressive manner, as the results keep arriving. This is a very important tool that enables us to aggregate constantly the information coming

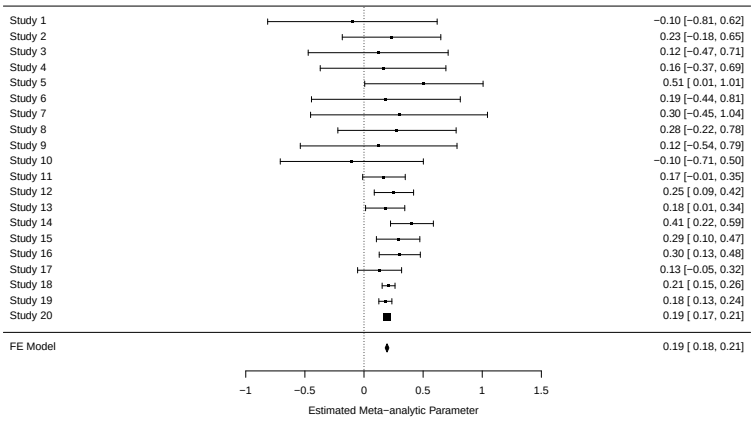


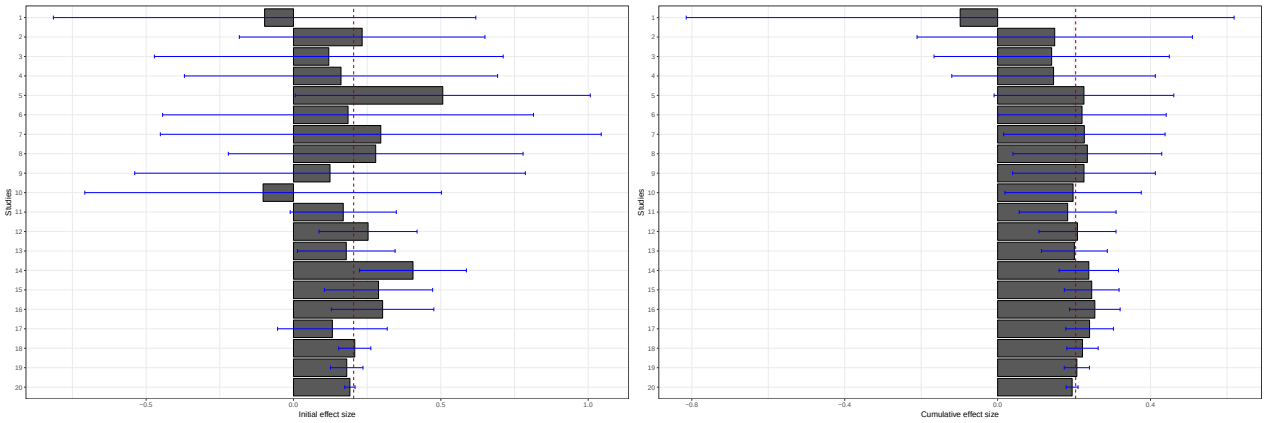
Figure 12.2: Example data set: forest plot

from different studies. Moreover, restrospectively, it helps us to assess when we would have reached enough precision so that we could have foregone an additional study. The way constantly updated meta-analysis works is simply by performing a new meta-analysis each time a new results pops up.

Example 12.3. Figure 12.3 shows how constantly updated meta-analysis works in our example.

```
cum.wmae.1 <- function(k,theta,sigma2){
  return(c(weighted.mean(theta[1:k],(1/sigma2[1:k])/(sum(1/sigma2[1:k]))),1/sum(1/sigma2[1:k])))
}

cum.wmae <- function(theta,sigma2){
  return(sapply(1:length(theta),cum.wmae.1,theta=theta,sigma2=sigma2))
}

cum.test <- as.data.frame(t(cum.wmae(data.meta$ES,data.meta$se.ES^2)))
colnames(cum.test) <- c('cum.ES','cum.var')
cum.test$id <- 1:nrow(cum.test)
cum.test$cum.se.ES <- sqrt(cum.test$cum.var)

  ggplot(data.meta, aes(x=forcats::fct_rev(as.factor(id)), y=ES)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=ES-qnorm((delta.2+1)/2)*se.ES, ymax=ES+qnorm((delta.2+1)/2)*se.ES), width=
      geom_hline(aes(yintercept=ES(param)), colour="#990000", linetype="dashed")+
      xlab("Studies")+
      ylab("Initial effect size")+
      theme_bw()+
      coord_flip()

  ggplot(cum.test, aes(x=forcats::fct_rev(as.factor(id)), y=cum.ES)) +
      geom_bar(position=position_dodge(), stat="identity", colour='black') +
      geom_errorbar(aes(ymin=cum.ES-qnorm((delta.2+1)/2)*cum.se.ES, ymax=cum.ES+qnorm((delta.2+1)/2)*cu
      geom_hline(aes(yintercept=ES(param)), colour="#990000", linetype="dashed")+
      xlab("Studies")+
      ylab("Cumulative effect size")+
      theme_bw()+
      coord_flip()
```

Figure 12.3 shows that combining several imprecise estimates might help you reach the same precision as running a larger experiment.



(a) Initial data set

(b) Constantly updated dataset

Figure 12.3: Constantly updated meta-analysis

For instance, cumulating the first 10 studies with a small sample size ($N = 100$), the meta-analytic effect is estimated at 0.2 ± 0.18. This is very close to the individual estimate obtained from the first estimate with a larger sample size (sample 11 on Figure 12.3, with $N = 1000$): 0.17 ± 0.18. Both estimates actually have the exact same precision (because they actually have the same sample size). The same is true when combining the first 17 studies. The meta-analytic effect is estimated at 0.24 ± 0.06, while the effect estimated using one unique RCT with a larger sample size (sample 18 on Figure 12.3, with $N = 10\hat{}\{4\}$) is 0.21 ± 0.05. Finally, the same result occurs when combining the first 19 studies. The meta-analytic effect is estimated at 0.21 ± 0.03, while the effect estimated using one unique RCT with a larger sample size (sample 20 on Figure 12.3, with $N = 10\hat{}\{5\}$) is 0.19 ± 0.02.

As a conclusion, constantly updated meta-analysis would have each time delivered the same result than the one found with a much larger study, rendering this additional study almost irrelevant. This is a very important result: beyond the apparent messiness of the first noisy estimates in Figures 12.1 and 12.2 lies an order that can be retrieved and made apparent using constantly updated meta-analysis. Sometimes, the answer is right there in front of our eyes, we just lack the ability to see it. Constantly updated meta-analysis serves as a binocular to magnify what is there. Think about how costly it woud be to run a very large study, just to find out that the we did not really need it because we had known the result all along.

12.1.4 Testing for the homogeneity of treatment effects

12.1.5 Meta-analysis when treatment effects are heterogeneous

12.1.6 Meta-regression

12.1.7 Why vote-counting does not work

12.2 Publication bias

12.2.1 Sources of publication bias

12.2.2 Detecting and correcting for publication bias

12.2.2.1 P-curving

12.2.2.2 PET-PEESE

12.2.2.3 Kasy and Andrews approach

12.2.2.4 Last approach

12.2.3 Vote counting and publication bias

Chapter 13

Bounds

Appendix A

Proofs

A.1 Proofs of results in Chapter 2

A.1.1 Proof of Theorem 2.3

In order to use Theorem 2.2 for studying the behavior of $\Delta^{\hat{Y}}_{WW}$, we have to prove that it is unbiased and we have to compute $\mathbb{V}[\Delta^{\hat{Y}}_{WW}]$. Let's first prove that the WW estimator is an unbiased estimator of TT:

Lemma A.1 (Unbiasedness of $\Delta^{\hat{Y}}_{WW}$)**.** *Under Assumptions 1.7, 2.3 and 2.4,*

$$\mathbb{E}[\Delta^{\hat{Y}}_{WW}] = \Delta^{Y}_{TT}.$$

Proof. In order to prove Lemma A.1, we are going to use a trick. We are going to compute the expectation of the WW estimator conditional on a given treatment allocation. Because the resulting estimate is independent of treatment allocation, we will have our proof. This trick simplifies derivations a lot and is really natural: think first of all the samples with the same treatment allocation, then average your results over all possible treatment allocations.

$$
\begin{aligned}
\mathbb{E}[\Delta^{\hat{Y}}_{WW}] &= \mathbb{E}[\mathbb{E}[\Delta^{\hat{Y}}_{WW}|\mathbf{D}]] \\
&= \mathbb{E}[\mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} Y_i D_i - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N} Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} Y_i D_i|\mathbf{D}] - \mathbb{E}[\frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N} Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\mathbb{E}[\sum_{i=1}^{N} Y_i D_i|\mathbf{D}] - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\mathbb{E}[\sum_{i=1}^{N} Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N}\mathbb{E}[Y_i D_i|\mathbf{D}] - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}\mathbb{E}[Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N}\mathbb{E}[Y_i D_i|D_i] - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}\mathbb{E}[Y_i(1-D_i)|D_i]] \\
&= \mathbb{E}[\frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} D_i\mathbb{E}[Y_i|D_i=1] - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N}(1-D_i)\mathbb{E}[Y_i|D_i=0]] \\
&= \mathbb{E}[\frac{\sum_{i=1}^{N} D_i}{\sum_{i=1}^{N} D_i}\mathbb{E}[Y_i|D_i=1] - \frac{\sum_{i=1}^{N}(1-D_i)}{\sum_{i=1}^{N}(1-D_i)}\mathbb{E}[Y_i|D_i=0]] \\
&= \mathbb{E}[\mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i|D_i=0]] \\
&= \mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i|D_i=0] \\
&= \Delta^{Y}_{TT}.
\end{aligned}
$$

The first equality uses the Law of Iterated Expectations (LIE). The second and fourth equalities use the linearity of conditional expectations. The third equality uses the fact that, conditional on $\mathbf{D}$, the number of treated and untreated is a constant. The fifth equality uses Assumption 2.4. The sixth equality uses the fact that $\mathbb{E}[Y_i D_i|D_i] = D_i\mathbb{E}[Y_i * 1|D_i = 1] + (1 - D_i)\mathbb{E}[Y_i * 0|D_i = 0]$. The seventh and ninth equalities use the fact that $\mathbb{E}[Y_i|D_i = 1]$ is a constant. The last equality uses Assumption 1.7. □

Let's now compute the variance of the WW estimator:

Lemma A.2 (Variance of $\Delta^{\hat{Y}}_{WW}$)**.** *Under Assumptions 1.7, 2.3 and 2.4,*

$$
\mathbb{V}[\Delta^{\hat{Y}}_{WW}] = \frac{1-(1-\Pr(D_i=1))^N}{N\Pr(D_i=1)}\mathbb{V}[Y_i^1|D_i=1] + \frac{1-\Pr(D_i=1)^N}{N(1-\Pr(D_i=1))}\mathbb{V}[Y_i^0|D_i=0].
$$

Proof. Same trick as before, but now using the Law of Total Variance (LTV):

$$
\begin{aligned}
\mathbb{V}[\hat{\Delta^Y_{WW}}] &= \mathbb{E}[\mathbb{V}[\hat{\Delta^Y_{WW}}|\mathbf{D}]] + \mathbb{V}[\mathbb{E}[\hat{\Delta^Y_{WW}}|\mathbf{D}]] \\
&= \mathbb{E}[\mathbb{V}[\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i D_i - \frac{1}{\sum_{i=1}^N (1-D_i)}\sum_{i=1}^N Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\mathbb{V}[\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i D_i|\mathbf{D}]] + \mathbb{E}[\mathbb{V}[\frac{1}{\sum_{i=1}^N (1-D_i)}\sum_{i=1}^N Y_i(1-D_i)|\mathbf{D}]] \\
&\quad + \mathbb{E}[\mathbb{C}[\frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i D_i, \frac{1}{\sum_{i=1}^N (1-D_i)}\sum_{i=1}^N Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\frac{1}{(\sum_{i=1}^N D_i)^2}\mathbb{V}[\sum_{i=1}^N Y_i D_i|\mathbf{D}]] + \mathbb{E}[\frac{1}{(\sum_{i=1}^N (1-D_i))^2}\mathbb{V}[\sum_{i=1}^N Y_i(1-D_i)|\mathbf{D}]] \\
&= \mathbb{E}[\frac{1}{(\sum_{i=1}^N D_i)^2}\mathbb{V}[\sum_{i=1}^N Y_i D_i|D_i]] + \mathbb{E}[\frac{1}{(\sum_{i=1}^N (1-D_i))^2}\mathbb{V}[\sum_{i=1}^N Y_i(1-D_i)|D_i]] \\
&= \mathbb{E}[\frac{1}{(\sum_{i=1}^N D_i)^2}\sum_{i=1}^N D_i\mathbb{V}[Y_i|D_i=1]] + \mathbb{E}[\frac{1}{(\sum_{i=1}^N (1-D_i))^2}\sum_{i=1}^N (1-D_i)\mathbb{V}[Y_i|D_i=0]] \\
&= \mathbb{V}[Y_i|D_i=1]\mathbb{E}[\frac{1}{\sum_{i=1}^N D_i}] + \mathbb{V}[Y_i|D_i=0]\mathbb{E}[\frac{1}{\sum_{i=1}^N (1-D_i)}] \\
&= \frac{1-(1-\Pr(D_i=1))^N}{N\Pr(D_i=1)}\mathbb{V}[Y_i^1|D_i=1] + \frac{1-\Pr(D_i=1)^N}{N(1-\Pr(D_i=1))}\mathbb{V}[Y_i^0|D_i=0].
\end{aligned}
$$

The first equality stems from the LTV. The second and third equalities stems from the definition of the WW estimator and of the variance of a sum of random variables. The fourth equality stems from Assumption 2.4, which means that the covariance across observations is zero, and from the formula for a variance of a random variable multiplied by a constant. The fifth and sixth equalities stems from Assumption 2.4 and from $\mathbb{V}[Y_i D_i|D_i] = D_i\mathbb{V}[Y_i * 1|D_i=1] + (1-D_i)\mathbb{V}[Y_i * 0|D_i=0]$. The seventh equality stems from $\mathbb{V}[Y_i|D_i=1]$ and $\mathbb{V}[Y_i|D_i=0]$ being constant. The last equality stems from the formula for the expectation of the inverse of a sum of Bernoulli random variables with at least one of them taking value one which is the case under Assumption 2.3. □

Using Theorem 2.2, we have:

$$
\begin{aligned}
2\epsilon &\leq 2\sqrt{\frac{1}{N(1-\delta)}\left(\frac{1-(1-\Pr(D_i=1))^N}{\Pr(D_i=1)}\mathbb{V}[Y_i^1|D_i=1] + \frac{1-\Pr(D_i=1)^N}{(1-\Pr(D_i=1))}\mathbb{V}[Y_i^0|D_i=0]\right)} \\
&\leq 2\sqrt{\frac{1}{N(1-\delta)}\left(\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{(1-\Pr(D_i=1))}\right)},
\end{aligned}
$$

where the second equality stems from the fact that $\frac{(1-\Pr(D_i=1))^N}{\Pr(D_i=1)}\mathbb{V}[Y_i^1|D_i=1] + \frac{\Pr(D_i=1)^N}{(1-\Pr(D_i=1))}\mathbb{V}[Y_i^0|D_i=0] \geq 0$. This proves the result.

A.1.2 Proof of Theorem 2.5

Before proving Theorem 2.5, let me state a very useful result: $\hat{WW}$ can be computed using OLS:

Lemma A.3 (WW is OLS)**.** *Under Assumption 2.3, the OLS coefficient β in the following regression:*

$$Y_i = \alpha + \beta D_i + U_i$$

is the WW estimator:

$$\begin{aligned}\hat{\beta}_{OLS} &= \frac{\frac{1}{N}\sum_{i=1}^{N}\left(Y_i - \frac{1}{N}\sum_{i=1}^{N} Y_i\right)\left(D_i - \frac{1}{N}\sum_{i=1}^{N} D_i\right)}{\frac{1}{N}\sum_{i=1}^{N}\left(D_i - \frac{1}{N}\sum_{i=1}^{N} D_i\right)^2} \\ &= \Delta_{WW}^{\hat{Y}}.\end{aligned}$$

Proof. In matrix notation, we have:

$$\underbrace{\begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} 1 & D_1 \\ \vdots & \vdots \\ 1 & D_N \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} \alpha \\ \beta \end{pmatrix}}_{\Theta} + \underbrace{\begin{pmatrix} U_1 \\ \vdots \\ U_N \end{pmatrix}}_{U}$$

The OLS estimator is:

$$\hat{\Theta}_{OLS} = (X'X)^{-1}X'Y$$

Under the Full Rank Assumption, $X'X$ is invertible and we have:

$$\begin{aligned}(X'X)^{-1} &= \begin{pmatrix} N & \sum_{i=1}^{N} D_i \\ \sum_{i=1}^{N} D_i & \sum_{i=1}^{N} D_i^2 \end{pmatrix}^{-1} \\ &= \frac{1}{N\sum_{i=1}^{N} D_i^2 - \left(\sum_{i=1}^{N} D_i\right)^2} \begin{pmatrix} \sum_{i=1}^{N} D_i^2 & -\sum_{i=1}^{N} D_i \\ -\sum_{i=1}^{N} D_i & N \end{pmatrix}\end{aligned}$$

For simplicity, I omit the summation index:

$$\begin{aligned}\hat{\Theta}_{OLS} &= \frac{1}{N\sum D_i^2 - \left(\sum D_i\right)^2} \begin{pmatrix} \sum D_i^2 & -\sum D_i \\ -\sum D_i & N \end{pmatrix} \begin{pmatrix} \sum Y_i \\ \sum Y_i D_i \end{pmatrix} \\ &= \frac{1}{N\sum D_i^2 - \left(\sum D_i\right)^2} \begin{pmatrix} \sum D_i^2 \sum Y_i - \sum D_i \sum_{i=1}^{N} Y_i D_i \\ -\sum D_i \sum Y_i + N \sum Y_i D_i \end{pmatrix}\end{aligned}$$

Using $D_i^2 = D_i$, we have:

$$\hat{\Theta}_{OLS} = \begin{pmatrix} \frac{(\sum D_i)(\sum Y_i - \sum Y_i D_i)}{(\sum D_i)(N - \sum D_i)} \\ \frac{N\sum Y_i D_i - \sum D_i \sum Y_i}{N\sum D_i - (\sum D_i)^2} \end{pmatrix} = \begin{pmatrix} \frac{\sum (Y_i D_i + Y_i(1-D_i)) - \sum Y_i D_i}{\sum (1-D_i)} \\ \frac{N^2}{N^2} \frac{\frac{1}{N}\sum Y_i D_i - \frac{1}{N}\sum D_i \frac{1}{N}\sum Y_i + \frac{1}{N}\sum D_i \frac{1}{N}\sum Y_i - \frac{1}{N}\sum D_i \frac{1}{N}\sum Y_i}{\frac{1}{N}\sum D_i - 2\left(\frac{1}{N}\sum D_i\right)^2 + \left(\frac{1}{N}\sum D_i\right)^2} \end{pmatrix}$$
$$= \begin{pmatrix} \frac{\sum Y_i(1-D_i)}{\sum (1-D_i)} \\ \frac{\frac{1}{N}\sum \left(Y_i D_i - D_i \frac{1}{N}\sum Y_i - Y_i \frac{1}{N}\sum D_i + \frac{1}{N}\sum D_i \frac{1}{N}\sum Y_i\right)}{\frac{1}{N}\sum \left(D_i - 2D_i \frac{1}{N}\sum D_i + \left(\frac{1}{N}\sum D_i\right)^2\right)} \end{pmatrix} = \begin{pmatrix} \frac{\sum Y_i(1-D_i)}{\sum (1-D_i)} \\ \frac{\frac{1}{N}\sum \left(Y_i - \frac{1}{N}\sum Y_i\right)\left(D_i - \frac{1}{N}\sum D_i\right)}{\frac{1}{N}\sum \left(D_i - \frac{1}{N}\sum D_i\right)^2} \end{pmatrix},$$

which proves the first part of the lemma. Now for the second part of the lemma:

$$\begin{aligned}
\hat{\beta}_{OLS} &= \frac{\sum Y_i D_i - \frac{1}{N}\sum D_i \sum Y_i}{\sum D_i \left(1 - \frac{1}{N}\sum D_i\right)} = \frac{\sum Y_i D_i - \frac{1}{N}\sum D_i \sum (Y_i D_i + (1-D_i)Y_i)}{\sum D_i \left(1 - \frac{1}{N}\sum D_i\right)} \\
&= \frac{\sum Y_i D_i \left(1 - \frac{1}{N}\sum D_i\right) - \frac{1}{N}\sum D_i \sum (1-D_i)Y_i}{\sum D_i \left(1 - \frac{1}{N}\sum D_i\right)} \\
&= \frac{\sum Y_i D_i}{\sum D_i} - \frac{\frac{1}{N}\sum (1-D_i)Y_i}{\left(1 - \frac{1}{N}\sum D_i\right)} \\
&= \frac{\sum Y_i D_i}{\sum D_i} - \frac{\frac{1}{N}\sum (1-D_i)Y_i}{\frac{1}{N}\sum (1-D_i)} \\
&= \frac{\sum Y_i D_i}{\sum D_i} - \frac{\sum (1-D_i)Y_i}{\sum (1-D_i)} \\
&= \Delta^{\hat{Y}}_{WW},
\end{aligned}$$

which proves the result. □

Now, let me state the most important lemma behind the result in Theorem 2.5:
Lemma A.4 (Asymptotic Distribution of the OLS Estimator)**.** *Under Assumptions 1.7, 2.3, 2.4 and 2.5, we have:*

$$\sqrt{N}(\hat{\Theta}_{OLS} - \Theta) \xrightarrow{d} \mathcal{N}\left(\begin{array}{c} 0 \\ 0 \end{array}, \sigma_{XX}^{-1} \mathbf{V_{xu}} \sigma_{XX}^{-1} \right),$$

with

$$\sigma_{XX}^{-1} = \begin{pmatrix} \frac{\Pr(D_i=1)}{\Pr(D_i=1)(1-\Pr(D_i=1))} & -\frac{\Pr(D_i=1)}{\Pr(D_i=1)(1-\Pr(D_i=1))} \\ -\frac{\Pr(D_i=1)}{\Pr(D_i=1)(1-\Pr(D_i=1))} & \frac{1}{\Pr(D_i=1)(1-\Pr(D_i=1))} \end{pmatrix}$$
$$\mathbf{V_{xu}} = \mathbb{E}[U_i^2 \begin{pmatrix} 1 & D_i \\ D_i & D_i \end{pmatrix}]$$

Proof.

$$\begin{aligned}\sqrt{N}(\hat{\Theta}_{OLS} - \Theta) &= \sqrt{N}((X'X)^{-1}X'Y - \Theta) \\ &= \sqrt{N}((X'X)^{-1}X'(X\Theta + U) - \Theta) \\ &= \sqrt{N}((X'X)^{-1}X'X\Theta + (X'X)^{-1}X'U) - \Theta) \\ &= \sqrt{N}(X'X)^{-1}X'U \\ &= N(X'X)^{-1}\frac{\sqrt{N}}{N}X'U\end{aligned}$$

Using Slutsky's Theorem, we can study both terms separately. Slutsky's Theorem states that if $Y_N \xrightarrow{d} y$ and $\text{plim}(X_N) = x$, then:

1. $X_N + Y_N \xrightarrow{d} x + y$
2. $X_N Y_N \xrightarrow{d} xy$
3. $\frac{Y_N}{X_N} \xrightarrow{d} \frac{x}{y}$ if $x \neq 0$

Using this theorem, we have:

$$\sqrt{N}(\hat{\Theta}_{OLS} - \Theta) \xrightarrow{d} \sigma_{XX}^{-1} xu,$$

Where σ_{XX}^{-1} is a matrix of constants and xu is a random variable.

Let's begin with $\frac{\sqrt{N}}{N}X'U \xrightarrow{d} xu$:

$$\frac{\sqrt{N}}{N}X'U = \sqrt{N}\begin{pmatrix} \frac{1}{N}\sum_N^{i=1} U_i \\ \frac{1}{N}\sum_N^{i=1} D_i U_i \end{pmatrix}$$

In order to determine the asymptotic distribution of $\frac{\sqrt{N}}{N}X'U$, we are going to use the vector version of the CLT:

If X_i and Y_i are two i.i.d. random variables with finite first and second moments, we have:

$$\sqrt{N}\begin{pmatrix} \frac{1}{N}\sum_{i=1}^{N} X_i - \mathbb{E}[X_i] \\ \frac{1}{N}\sum_{i=1}^{N} Y_i - \mathbb{E}[Y_i] \end{pmatrix} \xrightarrow{d} \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{V}\right),$$

where $\mathbf{V}$ is the population covariance matrix of X_i and Y_i.

We know that, under Assumption 1.7, both random variables have mean zero:

$$\begin{aligned}\mathbb{E}[U_i] &= \mathbb{E}[U_i|D_i = 1]\Pr(D_i = 1) + \mathbb{E}[U_i|D_i = 0]\Pr(D_i = 0) = 0 \\ \mathbb{E}[U_i D_i] &= \mathbb{E}[U_i|D_i = 1]\Pr(D_i = 1) = 0\end{aligned}$$

Their covariance matrix $\mathbf{V_{xu}}$ can be computed as follows:

$$\begin{aligned}\mathbf{V_{xu}} &= \mathbb{E}[\begin{pmatrix} U_i \\ UiD_i \end{pmatrix}\begin{pmatrix} U_i & UiD_i \end{pmatrix}] - \mathbb{E}[\begin{pmatrix} U_i \\ UiD_i \end{pmatrix}]\mathbb{E}[\begin{pmatrix} U_i & UiD_i \end{pmatrix}] \\ &= \mathbb{E}[\begin{pmatrix} U_i^2 & U_i^2 D_i \\ Ui^2 D_i & U_i^2 D_i^2 \end{pmatrix}] = \mathbb{E}[U_i^2\begin{pmatrix} 1 & D_i \\ D_i & D_i^2 \end{pmatrix}] = \mathbb{E}[U_i^2\begin{pmatrix} 1 & D_i \\ D_i & D_i \end{pmatrix}]\end{aligned}$$

Using the Vector CLT, we have that $\frac{\sqrt{N}}{N}X'U \xrightarrow{d} \mathcal{N}\left(\begin{array}{c} 0 \\ 0 \end{array} , \mathbf{V_{xu}} \right)$.

Let's show now that $\text{plim}N(X'X)^{-1} = \sigma_{XX}^{-1}$:

$$\begin{aligned}
N(X'X)^{-1} &= \frac{N}{N\sum_{i=1}^N D_i - \left(\sum_{i=1}^N D_i\right)^2} \begin{pmatrix} \sum_{i=1}^N D_i & -\sum_{i=1}^N D_i \\ -\sum_{i=1}^N D_i & N \end{pmatrix} \\
&= \frac{1}{N}\frac{1}{\frac{1}{N}\sum_{i=1}^N D_i - \left(\frac{1}{N}\sum_{i=1}^N D_i\right)^2} \begin{pmatrix} \sum_{i=1}^N D_i & -\sum_{i=1}^N D_i \\ -\sum_{i=1}^N D_i & N \end{pmatrix} \\
&= \frac{1}{\frac{1}{N}\sum_{i=1}^N D_i - \left(\frac{1}{N}\sum_{i=1}^N D_i\right)^2} \begin{pmatrix} \frac{1}{N}\sum_{i=1}^N D_i & -\frac{1}{N}\sum_{i=1}^N D_i \\ -\frac{1}{N}\sum_{i=1}^N D_i & 1 \end{pmatrix} \\
\text{plim}N(X'X)^{-1} &= \frac{1}{\text{plim}\frac{1}{N}\sum_{i=1}^N D_i - \left(\text{plim}\frac{1}{N}\sum_{i=1}^N D_i\right)^2} \begin{pmatrix} \text{plim}\frac{1}{N}\sum_{i=1}^N D_i & -\text{plim}\frac{1}{N}\sum_{i=1}^N D_i \\ -\text{plim}\frac{1}{N}\sum_{i=1}^N D_i & 1 \end{pmatrix} \\
&= \frac{1}{\Pr(D_i=1) - \Pr(D_i=1)^2} \begin{pmatrix} \Pr(D_i=1) & -\Pr(D_i=1) \\ -\Pr(D_i=1) & 1 \end{pmatrix} \\
&= \sigma_{XX}^{-1}
\end{aligned}$$

The fourth equality uses Slutsky's Theorem. The fifth equality uses the Law of Large Numbers (LLN): if Y_i are i.i.d. variables with finite first and second moments, $\text{plim}_{N\to\infty}\frac{1}{N}\sum_{i=1}^N Y_i = \mathbb{E}[Y_i]$.

In order to complete the proof, we have to use the Delta Method Theorem. This theorem states that:

$$\begin{aligned}
&\sqrt{N}\begin{pmatrix} \bar{X}_N - \mathbb{E}[X_i] \\ \bar{Y}_N - \mathbb{E}[Y_i] \end{pmatrix} \xrightarrow{d} \mathcal{N}\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{V}) \\
\Rightarrow &\sqrt{N}(g(\bar{X}_N, \bar{Y}_N) - g(\mathbb{E}[X_i], \mathbb{E}[Y_i]) \xrightarrow{d} \mathcal{N}(0, G'\mathbf{V}G)
\end{aligned}$$

where $G(u) = \frac{\partial g(u)}{\partial u}$ and $G = G(\mathbb{E}[X_i], \mathbb{E}[Y_i])$.

In our case, $g(xu) = \sigma_{XX}^{-1}xu$, so $G(xu) = \sigma_{XX}^{-1}$. The results follows from that and from the symmetry of σ_{XX}^{-1}. □

A last lemma uses the previous result to derive the asymptotic distribution of $W\hat{W}$:

Lemma A.5 (Asymptotic Distribution of $W\hat{W}$)**.** *Under Assumptions 1.7, 2.3, 2.4 and 2.5, we have:*

$$\sqrt{N}(\Delta_{WW}^{\hat{Y}} - \Delta_{TT}^{Y}) \xrightarrow{d} \mathcal{N}\left(0, \frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}\right).$$

Proof. In order to derive the asymptotic distribution of WW, I use first Lemma A.3 which implies that the asymptotic distribution of WW is the same as that of $\hat{\beta}_{OLS}$. Now, from Lemma A.4, we know that $\sqrt{N}(\hat{\beta}_{OLS} - \beta) \xrightarrow{d} \mathcal{N}(0, \sigma_\beta^2)$, where σ_β^2 is the lower diagonal term of $\sigma_{XX}^{-1}\mathbf{V_{xu}}\sigma_{XX}^{-1}$. Using the convention $p = \Pr(D_i = 1)$, we have:

$$\begin{aligned}
\sigma_{XX}^{-1}\mathbf{V_{xu}}\sigma_{XX}^{-1} &= \begin{pmatrix} \frac{p}{p(1-p)} & -\frac{p}{p(1-p)} \\ -\frac{p}{p(1-p)} & \frac{1}{p(1-p)} \end{pmatrix} \mathbb{E}[U_i^2 \begin{pmatrix} 1 & D_i \\ D_i & D_i \end{pmatrix}] \begin{pmatrix} \frac{p}{p(1-p)} & -\frac{p}{p(1-p)} \\ -\frac{p}{p(1-p)} & \frac{1}{p(1-p)} \end{pmatrix} \\
&= \frac{1}{(p(1-p))^2} \begin{pmatrix} p\mathbb{E}[U_i^2] - p\mathbb{E}[U_i^2 D_i] & p\mathbb{E}[U_i^2 D_i] - p\mathbb{E}[U_i^2 D_i] \\ -p\mathbb{E}[U_i^2] + \mathbb{E}[U_i^2 D_i] & -p\mathbb{E}[U_i^2 D_i] + \mathbb{E}[U_i^2 D_i] \end{pmatrix} \begin{pmatrix} p & -p \\ -p & 1 \end{pmatrix} \\
&= \frac{1}{(p(1-p))^2} \begin{pmatrix} p^2(\mathbb{E}[U_i^2] - \mathbb{E}[U_i^2 D_i]) & p^2(\mathbb{E}[U_i^2 D_i] - \mathbb{E}[U_i^2]) \\ p^2(\mathbb{E}[U_i^2 D_i] - \mathbb{E}[U_i^2]) & p^2\mathbb{E}[U_i^2] + (1-2p)\mathbb{E}[U_i^2 D_i] \end{pmatrix}
\end{aligned}$$

The final result comes from the fact that:

$$\begin{aligned}
\mathbb{E}[U_i^2] &= \mathbb{E}[U_i^2 | D_i = 1]p + (1-p)\mathbb{E}[U_i^2 | D_i = 0] \\
&= p\mathbb{V}[Y_i^1 | D_i = 1] + (1-p)\mathbb{V}[Y_i^0 | D_i = 0] \\
\mathbb{E}[U_i^2 D_i] &= \mathbb{E}[U_i^2 | D_i = 1]p \\
&= p\mathbb{V}[Y_i^1 | D_i = 1].
\end{aligned}$$

As a consequence:

$$\begin{aligned}
\sigma_\beta^2 &= \frac{1}{(p(1-p))^2} \left(\mathbb{V}[Y_i^1 | D_i = 1]p(p^2 - 2p + 1) + p^2(1-p)\mathbb{V}[Y_i^0 | D_i = 0] \right) \\
&= \frac{1}{(p(1-p))^2} \left(\mathbb{V}[Y_i^1 | D_i = 1]p(1-p)^2 + p^2(1-p)\mathbb{V}[Y_i^0 | D_i = 0] \right) \\
&= \frac{\mathbb{V}[Y_i^1 | D_i = 1]}{p} + \frac{\mathbb{V}[Y_i^0 | D_i = 0]}{1-p}.
\end{aligned}$$

□

Using the previous lemma, we can now approximate the confidence level of $\hat{WW}$:

$$\begin{aligned}
&\Pr(|\Delta_{WW}^{\hat{Y}} - \Delta_{TT}^{Y}| \leq \epsilon) = \Pr(-\epsilon \leq \Delta_{WW}^{\hat{Y}} - \Delta_{TT}^{Y} \leq \epsilon) \\
&= \Pr\left(-\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \leq \frac{\Delta_{WW}^{\hat{Y}} - \Delta_{TT}^{Y}}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \leq \frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) \\
&\approx \Phi\left(\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) - \Phi\left(-\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) \\
&= \Phi\left(\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) - 1 + \Phi\left(\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) \\
&= 2\Phi\left(\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)} + \frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}} \right) - 1.
\end{aligned}$$

As a consequence,

$$\delta \approx 2\Phi\left(\frac{\epsilon}{\frac{1}{\sqrt{N}}\sqrt{\frac{\mathbb{V}[Y_i^1|D_i=1]}{\Pr(D_i=1)}+\frac{\mathbb{V}[Y_i^0|D_i=0]}{1-\Pr(D_i=1)}}}\right)-1.$$

Hence the result.

A.2 Proofs of results in Chapter 3

A.2.1 Proof of Lemma 3.1

In matrix notation, we have:

$$\underbrace{\begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} 1 & D_1 \\ \vdots & \vdots \\ 1 & D_N \end{pmatrix}}_{X} \underbrace{\begin{pmatrix} \alpha \\ \beta \end{pmatrix}}_{\Theta} + \underbrace{\begin{pmatrix} U_1 \\ \vdots \\ U_N \end{pmatrix}}_{U}$$

and

$$\begin{pmatrix} D_1 \\ \vdots \\ D_N \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & R_1 \\ \vdots & \vdots \\ 1 & R_N \end{pmatrix}}_{R} \begin{pmatrix} \gamma \\ \tau \end{pmatrix} + \begin{pmatrix} V_1 \\ \vdots \\ V_N \end{pmatrix}$$

The IV estimator is:

$$\hat{\Theta}_{IV} = (R'X)^{-1}R'Y$$

Under Assumption 2.3, $R'X$ is invertible and we have (ommitting the summation index for simplicity):

$$\begin{aligned}(R'X)^{-1} &= \begin{pmatrix} N & \sum D_i \\ \sum R_i & \sum D_i R_i \end{pmatrix}^{-1} \\ &= \frac{1}{N\sum D_i R_i - \sum D_i \sum R_i}\begin{pmatrix} \sum D_i R_i & -\sum D_i \\ -\sum R_i & N \end{pmatrix}\end{aligned}$$

We have:

$$\hat{\Theta}_{IV} = \begin{pmatrix} \frac{\sum Y_i \sum D_i R_i - \sum D_i \sum Y_i R_i}{N\sum D_i R_i - \sum D_i R_i} \\ \frac{N\sum Y_i R_i - \sum R_i \sum Y_i}{N\sum D_i R_i - \sum D_i R_i} \end{pmatrix}$$

As a consequence, $\hat{\beta}_{IV}$ is equal to the ratio of two OLS estimators (Y_i on R_i and a constant and D_i on the same regressors) (see the proof of Lemma A.3 in section A.1.2, just after "Using $D_i^2 = D_i$"). We can use Lemma A.3 stating that the OLS estimator is the WW estimator to prove the result.