# EPE - Exercices
# The Two Fundamental Problems of Inference

Thibault Daly

February 18, 2017

**Abstract**

In this paper we will do a exercice of repdroduction. Indeed, this paper will be the solution of exercices on the Lectures 0. In order to do that we will use knitr and produce our results from Rstudio.

## 1 Assignements

1. Create a knitr file .Rnw
2. Generate the data with baseline parameter values
3. plot Figure 1 and Table 1
4. plot potential outcomes along yB as in the slides
5. Compute TTs, WW and SB in the sample
6. Compute BA and TB in the sample
7. Generate data without selection bias
8. Compute WW and SB
9. Compute placebo test for WW and BA
10. Generate data without time trend bias
11. Compute BA and TB
12. Compute placebo test
13. dowlnoad data from the paper you chose to analyze (use the readDTS function in R to import the .rds files)
14. On the data for the treatment arm, compute WW and BA if you can (if pre-treatment outcome exists)
15. Estimate Cheyshev's upper bound on precision in generated data
16. Estimate CLT-based approximation to sampling noise in generated data
17. Estimate bootstrapped approximation to sampling noise in generated data

18. Estimate Fisher-based approximation to sampling noise in generated data
19. Follow the same steps in the treatment arm of your RCT
20. Advanced: for those who fill like it, look at what happens when the treatment effect is in levels, not logs (Use Monte Carlos). CLT should perform less well in small samples.
21. dowlnoad data from the paper you chose to analyze (use the readDTS function in R to import the .rds files)
22. On the data for the treatment arm, compute WW and BA if you can (if pre-treatment outcome exists)
23. Estimate Cheyshev's upper bound on precision in generated data
24. Estimate CLT-based approximation to sampling noise in generated data
25. Estimate bootstrapped approximation to sampling noise in generated data
26. Estimate Fisher-based approximation to sampling noise in generated data
27. Follow the same steps in the treatment arm of your RCT

## 1.1 Create a knitr file .Rnw

First step we created a knitr file which is a R file with a combination of latex.

## 1.2 Generate the data with baseline parameter values

Here we are asked to generate a data with a certain framework that was well deteled in class :

$$y_i^B = \mu_i + U_i^B \textit{ with } \mu_i \sim \mathcal{N}(\mu,\, \sigma^2_\mu) \textit{ and } U_i^B \sim \mathcal{N}(0,\, \sigma^2_U)$$

We then select the values of the parameters :

```
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
```

In order to check the correctness of the computation we take the same random sample as the professor thanks to set.seed() And we then simulate a random sample following normals.

```
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 <- ifelse(YB<=param["barY"],1,0)
```

### 1.3 plot Figure 1 and Table 1

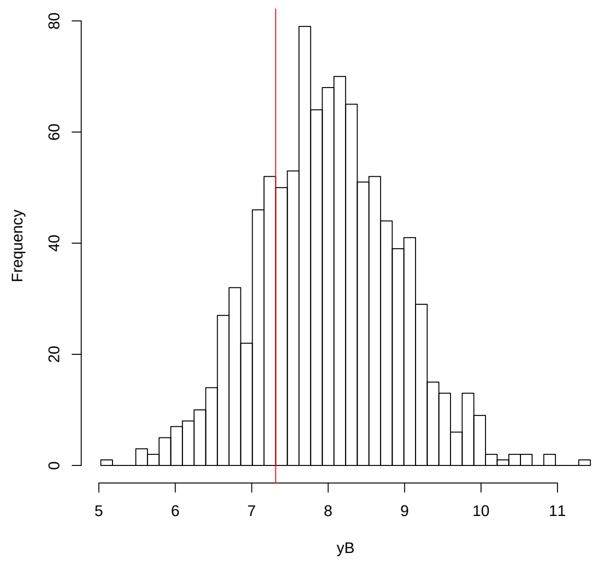

Figure 1: Histogram of $y_B$

You can see on Figure 1 a histogram of $y_i^B$ with the red line indicating the cutoff point: $\bar{y} = \ln(\bar{Y})$ =7.3132204.So on the left side of the cutoff value the sample is treated. We then design a table to see excatly how many people are treated and how much are not.

|   | Ds |
|---|---|
| 0 | 771 |
| 1 | 229 |

Table 1: Treatment allocation with sharp cutoff rule

### 1.4 plot potential outcomes along yB as in the slides

In order to plot the potential outcomes we first generate a data perfectly matching what we saw in class and here is the plot :

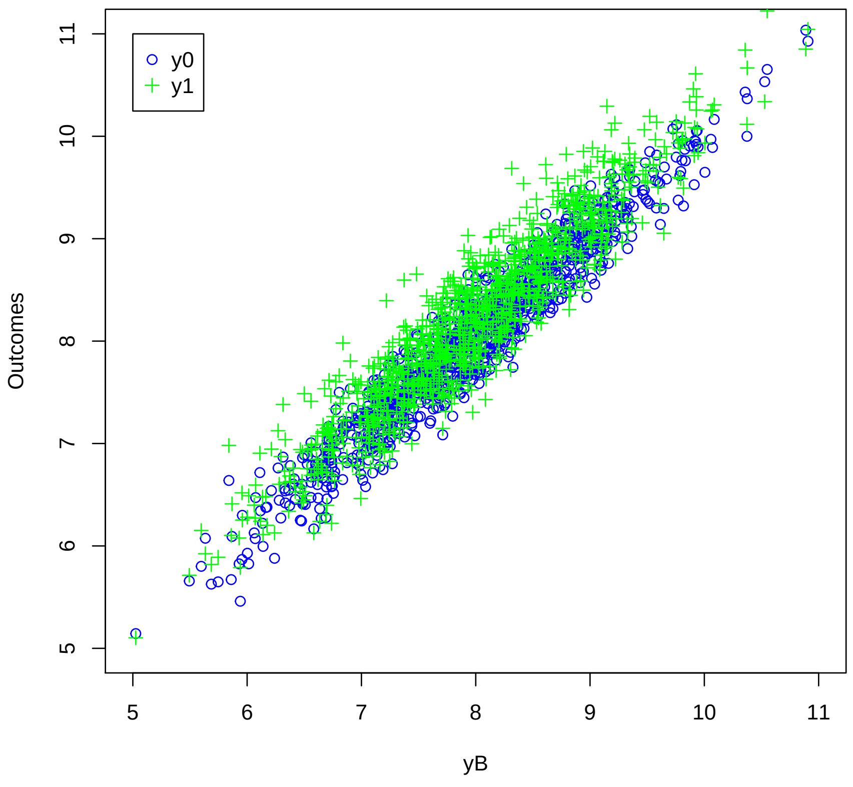

Figure 2: Potential outcomes

### 1.5 Compute TTs, WW and SB in the sample

- The average Treatment effect on the Treated in the sample (TT) :

$$\begin{aligned}\Delta^Y_{TT_s} &= \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i D_i - \frac{1}{\sum_{i=1}^N D_i}\sum_{i=1}^N Y_i^0 D_i \\ &= 0.168\end{aligned}$$

- The with/without comparison in the sample (WW) :

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

In our sample we get the following value for the WW :

$$\begin{aligned}\Delta^{\hat{Y}}_{WW} &= \frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} Y_i D_i - \frac{1}{\sum_{i=1}^{N}(1-D_i)}\sum_{i=1}^{N} Y_i(1-D_i)\\ &= -1.308\end{aligned}$$

- Selection bias (SB) :

$$\begin{aligned}\Delta^{Y}_{SB} &= \Delta^{Y}_{WW} - \Delta^{Y}_{TT}\\ &= \mathbb{E}[Y_i^1|D_i=1] - \mathbb{E}[Y_i^0|D_i=0] - (\mathbb{E}[Y_i^1|D_i=1] - \mathbb{E}[Y_i^0|D_i=1])\\ &= \mathbb{E}[Y_i^0|D_i=1] - \mathbb{E}[Y_i^0|D_i=0]\\ &= -1.476\end{aligned}$$

## 1.6 Compute BA and TB in the sample

- The before/after comparison (BA): The Before/After estimator compares the expected outcomes on the treated group before and after the treatment in the population

$$\begin{aligned}\Delta^{Y}_{BA} &= \mathbb{E}[Y_i|D_i=1] - \mathbb{E}[Y_i^B|D_i=1]\\ \Delta^{\hat{Y}}_{BA} &= \frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} Y_i D_i - \frac{1}{\sum_{i=1}^{N} D_i}\sum_{i=1}^{N} Y_i^B D_i\\ &= 0.267\end{aligned}$$

So the BA in the sample is $\Delta^{\hat{Y}}_{BA} = 0.267$.

- Time Trend Bias (TB):

$$\begin{aligned}\Delta^{Y}_{TB} &= \Delta^{Y}_{BA} - \Delta^{Y}_{TT}\\ &= \mathbb{E}[Y_i^1|D_i=1] - \mathbb{E}[Y_i^B|D_i=1] - (\mathbb{E}[Y_i^1|D_i=1] - \mathbb{E}[Y_i^0|D_i=1])\\ &= \mathbb{E}[Y_i^0|D_i=1] - \mathbb{E}[Y_i^B|D_i=1]\\ &= 0.099\end{aligned}$$

## 1.7 Compute placebo tests for WW and BA

A placebo test looks for eventual effects where we think their should be none,

We can perfom the placebo test that applies the $WW$ estimator to the untreated if $\Delta^{y}_{BA|D=0}\hat{} = 0$ for the untreated sample.

$$\Delta^{y}_{BA|D=0}\hat{} = 0.043$$

In our case $\Delta^{y}_{BA|D=0}\hat{} \neq 0$ hence we the placebo is not validated.

## 1.8 Generate data without selection bias

No Selection Bias in The Model Used in the Simulations, hence $\rho = 0 and \sigma^2_\mu = 0$

```
set.seed(1234)
N <-1000
mu <- rnorm(N,param["barmu"],sqrt(param["sigma2mu"]))
UB <- rnorm(N,0,sqrt(param["sigma2U"]))
yB <- mu + UB
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)
```

## 1.9 Compute WW and SB

If their is no selection bias, we get that WW identifies the TT parameters, $\Delta^Y_{WW} = \Delta^Y_{TT}$

$$\Delta^{\hat{y}}_{WW} = 0.199 \approx \Delta^{\hat{y}}_{TT} = 0.168$$
$$\Delta^{\hat{y}}_{SB} = 0.199 - 0.168 = 0.031$$

## 1.10 Compute placebo tests for WW and BA

In the absence of the treatment, no treatment effect should be detected before the program is implemented

$$\begin{aligned}\Delta^{Y^B}_{WW} &= \mathbb{E}[Y^B_i|D_i = 1] - \mathbb{E}[Y^B_i|D_i = 0]\\ &= 0\end{aligned}$$

In our sample we get $\Delta^{\hat{Y^B}}_{WW} = 7.95 - 7.99 = -0.04$ so here the data that we generated still contains time trend bias so the placebo test is not valid.

But we can perfom the placebo test that applies the $BA$ estimator to the untreated.

$$\Delta^{\hat{y}}_{BA|D=0} = 0.064$$

## 1.11 Generate data without time trend bias

In order to generate data wihtout time trend bias we need to change and fix to zero the following factors: $\delta = 0$ and $\rho = 0$

```
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",
```

```
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)
```

## 1.12 Compute BA and TB

The before/after comparison (BA): $\Delta^{\hat{y}}_{BA} = 0.173$.

Time Trend Bias (TB): we already computed $\Delta^{y}_{TT_s}$,

$$\Delta^{\hat{y}}_{TB} = \Delta^{\hat{y}}_{BA} - \Delta^{y}_{TT_s} = 0.173 - 0.168 = 0.005$$

In the data by construction their is no time trend so its normal that we get a value close to 0.

## 1.13 Compute placebo test

We can perfom the placebo test that applies the $BA$ estimator to the untreated, $\Delta^{y}_{BA|D=0}\hat{} = 0.007 \approx 0$ since their is not time trend in this data.

## 1.14 dowlnoad data from the paper you chose to analyze (use the readDTS function in R to import the .rds files)

```
data <- readRDS("C:/Users/xavierdaly/Desktop/LAST YEAR OF SCHOOL/SEMESTRE 10/Econometrics of Pro
data
```

## 1.15 On the data for the treatment arm, compute WW and BA if you can (if pre-treatment outcome exists)

The with/without comparison in the sample (WW) :

$$
\begin{aligned}
\Delta^{\hat{Y}}_{WW} &= \frac{1}{\sum_{i=1}^{N} D_i} \sum_{i=1}^{N} Y_i D_i - \frac{1}{\sum_{i=1}^{N}(1-D_i)} \sum_{i=1}^{N} Y_i(1-D_i) \\
&= -1.366
\end{aligned}
$$

The before/after comparison (BA):

$$\Delta^{\hat{y}}_{BA} = 0.173.$$

## 1.16 Estimate Cheyshev's upper bound on precision in generated data

First we write the function in R that will compute the Cheyshev's upper bound :

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

Now we generate ou usual sample an keep the same seed in order to check our result with the one obtained in the lectures.

```r
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, here are the result of the Chebyshev upper bound, $2\hat{\epsilon}$ =1.39.

## 1.17 Estimate CLT-based approximation to sampling noise in generated data

In this subsection we will compute the CLT-based approximation in the same generated data, first we create a function in R to compute this value :

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

Here is the result of the computation of our function, $2\hat{\bar{\epsilon}}$ =0.36.

### 1.18 Estimate bootstrapped approximation to sampling noise in generated data

In order to estimate the sampling noise with bootstrapped method on our generated data we replicate the small algorthim that was decine and explian in class and run it 500 times:

```r
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)
```

After the end of the algorthim the approximation of the sampling noise is, 0.35.

### 1.19 Estimate Fisher-based approximation to sampling noise in generated data

```r
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)
```

After running the code wich is prety similar as computing a bootstrap we have the approximation of sampling noise with the Fisher-based approximation, 0.366.