WEB MATERIAL

Simulation as a Tool for Teaching and Learning Epidemiologic Methods

Jacqueline E. Rudolph, Matthew P. Fox, and Ashley I. Naimi

Table of Contents

WEB APPENDIX
Software Code

Below, we include the code that can be used to carry out the examples discussed in the main text and above. These R programs can also be found on GitHub.

Non-differential misclassification

```
#################################################################################################################
#
# Purpose: To demonstrate one way to simulate non-differential misclassification (NDM) scenarios and
#          to estimate bias under those scenarios
#
# Author: Jacqueline E. Rudolph
#
# Required packages: ggplot2, tidyverse
#
# Last Updated: 15 Oct 2020
#
#################################################################################################################

##Load in necessary packages

packages <- c("tidyverse" ,"ggplot2")
for (package in packages) {
  library(package, character.only=T)
}


##Set up data set to store results
```

```
results <- data.frame(
  rep=NA,
  delta1=NA,
  delta2=NA,
  delta3=NA,
  delta4=NA,
  delta5=NA,
  stringsAsFactors = FALSE
)


# Run simulation ----------------------------------------------------------

reps <- 10000 #The number of simulations
n <- 1000     #Sample size within each simulation

#We carry out the simulations using the following function
simloop <- function(r, n) {

  set.seed(r)
  results$rep <- r

  #Create 2 continuous confounders (M1 and M2)
  #Both are log normally distributed: the corresponding normal distributions have mean 0 and s.d. 1
  m1 <- rlnorm(n, meanlog=0, sdlog=1)
  m2 <- rlnorm(n, meanlog=0, sdlog=1)

  #Create binary exposure, P(A=1)=0.5
  #A affected by M1 and M2: both with log(OR) of log(2)
  a <- rbinom(n, 1, 1/(1 + exp(-(-log(1/0.5 - 1) + log(2)*m1 + log(2)*m2))))
```

```
#Continuous outcome affected by exposure and confounders
#Normally distributed with mean determined by formula below and s.d. 6
#Intercept of 10; mean difference of 2 for A, M1, and M2
y <- rnorm(n, mean=(10 + 2*a + 2*m1 + 2*m2), sd=6)

#Exposure with NDM
#We select 15% of the sample to have misclassified A
error <- rnorm(n) #We use "error" to determine misclassification in anticipation of dependency below
a.err <- a
  a.err <- ifelse(a==1 & error > qnorm(0.85), 0, a.err) #Set sensitivity to 0.85
  a.err <- ifelse(a==0 & error > qnorm(0.95), 1, a.err) #Set specificity to 0.95

#Induce NDM of M1 that is (1) independent of and (2) dependent with the error in exposure
m1.indep <- m1 + 0.9*runif(n, min=-1, max=1) #Mismeasured M1, independent of error in A
m1.dep  <- m1 + 0.9*error #Mismeasured M1, dependent with error in A (because both depend on "error")

#Scenario 1: Linear model under no misclassification
results$delta1 <- summary(glm(y ~ a + m1 + m2, family=gaussian(link="identity")))$coefficients[2]

#Scenario 2: Linear model under NDM of binary exposure only
results$delta2 <- summary(glm(y ~ a.err + m1 + m2, family=gaussian(link="identity")))$coefficients[2]

#Scenario 3: Linear model under NDM of continuous confounder only
results$delta3 <- summary(glm(y ~ a + m1.dep + m2, family=gaussian(link="identity")))$coefficients[2]

#Scenario 4: Linear model under NDM of exposue and confounder, with independent errors
results$delta4 <- summary(glm(y ~ a.err + m1.indep + m2, family=gaussian(link="identity")))$coefficients[2]

#Scenario 5: Linear model under NDM of exposure and confounder, with dependent errors
results$delta5 <- summary(glm(y ~ a.err + m1.dep + m2, family=gaussian(link="identity")))$coefficients[2]
```

```
  return(results)
}

##Run the above function "reps" times, where reps is the number of simulations

all.res <- lapply(1:reps, function(x) {simloop(x, n)})
all.res <- do.call(rbind, all.res)


# Summarize ------------------------------------------------------------

  #Proportion of simulations that had an error in the mean difference greater than the null
  pos.error2 <- sum((all.res$delta2 - 2) > 0)/reps #Comparing scenario 2 to scenario 1
  pos.error3 <- sum((all.res$delta3 - 2) > 0)/reps #Comparing scenario 3 to scenario 1
  pos.error4 <- sum((all.res$delta4 - 2) > 0)/reps #Comparing scenario 4 to scenario 1
  pos.error5 <- sum((all.res$delta5 - 2) > 0)/reps #Comparing scenario 5 to scenario 1

  #Get the average mean difference across simulations for each scenario
  summ.res <- all.res %>%
    summarise(avg.delta1=mean(delta1),
              avg.delta2=mean(delta2),
              avg.delta3=mean(delta3),
              avg.delta4=mean(delta4),
              avg.delta5=mean(delta5)) %>%
    #Compute bias by comparing the average mean differences to the true mean difference of 2
    mutate(bias2=avg.delta2 - 2,
           bias3=avg.delta3 - 2,
           bias4=avg.delta4 - 2,
           bias5=avg.delta5 - 2)
```

```
# Visualize ------------------------------------------------------------

#Set formatting options

thm <- theme_classic() +
  theme(
    #Format text
    axis.title = element_text(family="Helvetica", size=16, color="black"),
    legend.text = element_text(family="Helvetica", size=16, color="black", margin=margin(t=0.25,b=0.25, unit="lines")),
    legend.title = element_text(family="Helvetica", size=16, color="black"),
    axis.text = element_text(family="Helvetica", size=16, color="black"),

    #Format axes
    axis.line = element_line(size=0.75),
    axis.ticks = element_line(size=0.75),

    #Format legend
    legend.title.align = 0.5,
    legend.position = c(0.85, 0.8),
    legend.background = element_rect(fill = "transparent", colour = NA),
    legend.key = element_rect(fill = "transparent", colour = NA),
    legend.direction="vertical",
    legend.box.background = element_rect(colour = "black", size=0.75),

    #Add space around plot
    plot.margin = unit(c(2, 2, 2, 2), "lines")
  )

#Compare results of Scenarios 1, 2, and 5
    #Define custom legend labels
```

```
cols<-c("None"="solid", "Exposure"="dashed", "Exposure & \nConfounder"="dotted")

pdf("./missclass_fig.pdf", height=6, width=9)
ggplot(all.res) + thm +
  #Plot labels
  labs(x="\n Mean Difference", y="Density \n") +

  #Generate plots
  stat_density(aes(delta1, linetype="None"), geom="line", size=0.75) +
  stat_density(aes(delta2, linetype="Exposure"), geom="line", size=0.75) +
  stat_density(aes(delta5, linetype="Exposure & \nConfounder"), geom="line", size=0.75) +

  #Create custom legend
  scale_linetype_manual(name="Misclassified \n Variable(s)", values=cols) +
  annotate(geom="segment", x=3.55, xend=4.63, y=0.88, yend=0.88, size=0.75, color="black") +

  #Define axis options
  scale_x_continuous(expand=c(0, 0), limits=c(-1, 5), breaks=c(-1, 0, 1, 2, 3, 4, 5)) +
  scale_y_continuous(expand=c(0,0), limits=c(0,1))
dev.off()
```

Defining a p-value

```
#################################################################################################
#
# Purpose: To show how simulation can be used to demonstrate the definition of a p-value
#
# Author: Jacqueline E. Rudolph
#
# Required packages: ggplot2, ggpubr
#
# Last Updated: 15 Oct 2020
#
#################################################################################################

##Load in necessary packages

packages <- c("ggplot2", "ggpubr")
for (package in packages) {
  library(package, character.only=T)
}


# Run simulation ----------------------------------------------------

reps <- 10000      #Number of permutations
n <- 1000          #Number of individuals in each sample

set.seed(123)
rd <- rep(NA,reps)  #Vector to hold risk difference results
```

```
 #Simulate randomized treatment, with P(X=1)=0.5
 x <- rbinom(n, 1, 0.5)

 #Simulate binary outcome affected by treatment
 #P(Y=1)=0.5 and the RD for X is 0.05
 p.y <- 0.5 + 0.05*x - 0.05*0.5
 y <- rbinom(n, 1, p.y)

 #Estimate the RD for X and its p-value
 res <- glm(y ~ x, family=gaussian(link="identity"))
 rd_est <- summary(res)$coefficients[2]
 rd_pvalue <- summary(res)$coefficients[8]

 #Estimate the distribution of RDs under the null hypothesis that RD=0
  #Randomly assign a new exposure to each individual, still with P(X'=1)=0.5
  #For this new exposure, estimate the RD on the original outcome
 for (i in 1:reps){
  #Draw new randomized exposure
  x_new <- rbinom(n, 1, 0.5)

  #Using new exposure and original outcome, estimate the RD and save its value in the rd vector
  rd[i] <- summary(glm(y ~ x_new, family=gaussian(link="identity")))$coefficients[2]
 }


# Summarize ------------------------------------------------------------

 #Calculate the one-sided  p-value, by taking the proportion of reps
 #that had a RD greater than the RD from the original sample
 prop1 <- sum(rd > rd_est)/reps
```

```
  #Calculate the two-sided p-value, by taking the proportion of reps
  #that had a RD more extreme than the RD from the original sample
    #In other words, the absolute value of the RD was greater than the original RD
  prop2 <- sum(abs(rd) > rd_est)/reps


# Visualize ------------------------------------------------------------

#Set formatting options

thm <- theme_classic() +
  theme(
    #Format text
    axis.title = element_text(family="Helvetica", size=16, color="black"),
    axis.text = element_text(family="Helvetica", size=16, color="black"),

    #Use tag to label paneled plots
    plot.tag.position="topleft",
    plot.tag=element_text(family="Helvetica", size=16, color="black", margin=margin(r=10, b=20)),

    #Format axis
    axis.line = element_line(size=0.75),
    axis.ticks = element_line(size=0.75),

    #Add space around plot
    plot.margin = unit(c(1, 1, 1, 1), "lines"),
  )

#Plot one-sided p-value

    p1 <- ggplot() + thm +
```

```
    #Plot labels
    labs(tag="A)", x="\nRisk Difference", y="Density\n") +

    #Generate plots
    geom_histogram(aes(rd, stat(density)), bins=36, color="gray", fill="lightgray") +
    geom_vline(xintercept=rd_est, color="black", size=0.75, linetype="dashed") +

    #Define axis options
    scale_x_continuous(expand=c(0, 0), limits=c(-0.12, 0.12)) +
    scale_y_continuous(expand=c(0, 0), limits=c(0, 15))

#Plot two-sided p-value

  p2 <- ggplot() + thm +
    labs(tag="B)", x="\nAbsolute Value of Risk Difference", y="Density\n") +
    geom_histogram(aes(abs(rd), stat(density)), bins=36, color="gray", fill="lightgray") +
    geom_vline(xintercept=rd_est, color="black", size=0.75, linetype="dashed") +
    scale_x_continuous(expand=c(0, 0), limits=c(-0.12, 0.12)) +
    scale_y_continuous(expand=c(0, 0), limits=c(0, 30))

  pdf("./pvalue_fig.pdf", height=5, width=10)
  ggarrange(p1, p2, ncol=2)
  dev.off()
```