Designing Simulation Studies

Ashley I Naimi

Spring 2024

Contents

1 The Aims of a Simulation Study

So far in the course, we've mostly covered technical ingredients needed to conduct a simulation study. These include things such writing functions, deploying loops, and using seeds. In this section, we're going to take a look at some more general, less technical items that need to be considered when designing and implementing a simulation study.

Much of this short course is motivated by an excellent article on conducting simulation studies by Morris et al. (2019). In this article, the authors outline the ADEMP framework for motivating a simulation study, which stands for:

- Aims
- Data Generating Mechanisms
- Estimands
- Methods
- Performance Measures

We'll cover these next, starting with the **aims** of a simulation study. There are a wide variety of reasons on why one might conduct a simulation study. Among these include:

- checking whether a given algebraic solution or new code to deploy a particular method works as expected.
- assessing the finite-sample properties of methods whose validity has been established using asymptotic approximations.
- comparing two or more different statistical methods under identical simulated conditions.
- calculating sample sizes or power for a given study design under known conditions.
- unpacking a particular method to better understand it's underlying logic.
- understanding the importance of certain assumptions on the validity of a particular method.

Let's provide some examples of simulation studies motivated by some of these aims.

1.1 Understanding p-values

This example is less of a simulation study and more of a simulation illustration. The primary aim of this example is to demonstrate some of the underlying logic of a p-value. P-values are notoriously difficult to understand and interpret (**?**), and led to a considerable and voluminous literature on the topic. We can demonstrate some basic ideas with a very simple simulated dataset.

The following two-by-two table demonstrates some simple data we can do this with:

```
rct_data <- matrix(
  c(53,193,139,350),
  ncol=2,
  byrow=T)

colnames(rct_data) <- c("event","nonevent")
rownames(rct_data) <- c("exposed","unexposed")
rct_data <- as.table(rct_data)
rct_data
```

```
##           event nonevent
## exposed      53      193
## unexposed   139      350
```

We can estimate the risk ratio, defined as the ratio of the probability of the outcome in the exposed versus unexposed:

```
risk_ratio <- (rct_data[1, 1]/sum(rct_data[1, ]))/(rct_data[2,
    1]/sum(rct_data[2, ]))
round(risk_ratio, 2)
```

```
## [1] 0.76
```

We can also use standard equations to obtain an estimate of the standard error for this risk ratio:

```
SE_lnRR <- sqrt((1/rct_data[1, 1] - 1/sum(rct_data[1, ])) + (1/rct_data[2,
    1] - 1/sum(rct_data[2, ])))
```

Using this standard error and risk ratio, we can construct a p-value using a standard z-test:

```
z <- (log(risk_ratio) - 0)/SE_lnRR
round(2 * pnorm(-abs(z)), 4)
```

```
## [1] 0.0498
```

This p-value suggests the probability of observing a risk ratio of 0.76 or larger (in absolute value) if there were no actual association between the exposure and the outcome is 0.0498. Unfortunately, this example doesn't shed much intuitive light on what's happening here.

Instead, we can break this procedure into several underlying steps. First, we'll construct a long dataset out of the table:

```
rct_data
```

```
##            event nonevent
## exposed       53       193
## unexposed    139       350
```

```
rct_data_long <- rbind(matrix(rep(c(1, 1), rct_data[1, 1]), ncol = 2),
    matrix(rep(c(0, 1), rct_data[2, 1]), ncol = 2), matrix(rep(c(1,
        0), rct_data[1, 2]), ncol = 2), matrix(rep(c(0, 0), rct_data[2,
        2]), ncol = 2))
nrow(rct_data_long)
```

```
## [1] 735
```

```
# re-shuffle rows
rct_data_long <- data.frame(rct_data_long[sample(nrow(rct_data_long)),
    ])
names(rct_data_long) <- c("X", "Y")
```

Our new dataset looks like this:

```
head(rct_data_long)
```

```
##   X Y
## 1 0 0
## 2 0 1
## 3 0 0
## 4 0 1
## 5 1 0
## 6 0 0
```

The first thing we'll do with this dataset is based on the assumption that there is **no effect** of the exposure on the outcome (the null hypothesis). If this is the case, then it follows that shuffling around (or permuting) the expsoure and re-estimating the risk ratio every time we shuffle would give us a valid distribution of the effect around the null. For example, we can pick (randomly) the exposure value for observation 732 and switch that value with observation 4.

Doing this randomly for each observation would give as a new dataset in which everyone's exposure value was switched, but there outcome remained the same.

This is where the simulation can come in useful. We can construct a variety of different datasets .

If we did this re-shuffling and re-estimation multiple times, we'd get a distribution of risk ratios that looked like this:

```
set.seed(123)

rr_permuted <- NULL

permutations <- 20000

for(i in 1:permutations){

  permuted <- rct_data_long
```

```
  permuted$X <- permuted$X[sample(length(permuted$X))] # shuffle the exposure, N = 735

  res <- log(mean(subset(permuted,X==1 )$Y)/mean(subset(permuted,X==0)$Y)) # recalculate RR

  rr_permuted <- rbind(rr_permuted,res)

}

rr_permuted <- data.frame(rr_permuted)
names(rr_permuted) <- "estimates"
```

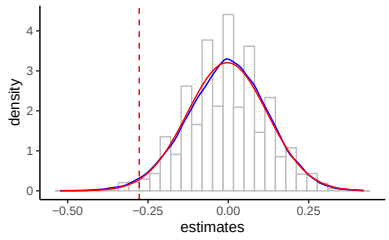

Figure 1: Distribution of log risk ratios after 2,000 random permutations of the exposure variable in the 2x2 table data above. The solid blue density curve represents a nonparametric kernel density estimate of the distribution. The solid red density curve represents a normal density estimate of the distribution. The dashed red vertical line indicates the value of the log risk ratio estimated in the original unpermuted data.

This permutation procedure gives us a critical component of a significance test: **the distribution of the estimates under the null.** It turns out, we can compute the p-value directly from this distribution.

There are a total of 2×10^4 estimates. How many of them are the same as or "more extreme" than the one we estimated in the actual data? We can compute this easily:

```
sum(rr_permuted$estimate <= log(risk_ratio))
```

```
## [1] 366
```

Dividing the number of estimates that are as or more extreme than the original risk ratio by 2×10^4 gives us a one-sided p-value:

```
sum(rr_permuted$estimate <= log(risk_ratio))/permutations
```

```
## [1] 0.0183
```

To get a two-sided test, we simply take the absolute values of both the original risk ratio and each estimate obtained in the permutation test, and repeat the comparison. Note that we have to change the direction of the "less than" sign for this to work:

```
sum(abs(rr_permuted$estimate) >= abs(log(risk_ratio)))/permutations
```

```
## [1] 0.02825
```

1.2 Non-Collapsibility of the OR

The next example will look at the impact of non-collapsiblility of the odds ratio (Greenland et al., 1999, Greenland (2005),Pang et al. (2013a)). The odds ratio is a non-collapsible measure of association, and for this reason its use in epidemiology is somewhat controversial (Pang et al., 2013b, Kaufman2010a). Non-collapsibility is a mathematical property of the odds ratio that results from Jensen's inequality (the average of a non-linear function does not equal the function its average; see Greenland and Pearl (2011)), and has confused many a statistician and epidemiologist (Greenland et al., 1999; Hernán et al., 2011).

In simple terms, non-collapsibility of the OR will be apparent when estimating an adjusted exposure-outcome association using a conditional and marginal approach (using standard logistic regression for the conditional approach, and IP-weighting or marginal standardization for the marginal approach). For example, in Figure 2,

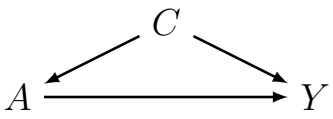

Figure 2: Simple confounding triangle, with exposure A, confounder C, and outcome Y.

we can adjust for the confounding effects of C conditionally using standard regression model:

$$\text{logit}\, P(Y = 1 \mid A, C) = \beta_0 + \beta_1 A + \beta_2 C,$$

and interpreting β_1 as the conditionally adjusted effect. Or, we can output predicted probabilities from this model for each person in the sample under two conditions: $A = 1$ and $A = 0$. We can then average these two probabilities over the sample, and compare the odds for the $A = 1$ to the odds for the $A = 0$. To explore the impact of noncollapsibility, we can simulate data following the causal relation in Figure 2.

Noncollapsibility is less of an issue when the outcome is rare. Often, the threshold for defining "rarity" is taken to be $\lessapprox 10\%$. But how valid is this $\lessapprox 10\%$ cutoff? We can answer this using Monte Carlo simulation. To do this, let's simulate data following the causal relation in Figure 2.

```
expit <- function(a){1/(1+exp(-a))}

set.seed(123)

n = 500
# simulate confounder from a normal distribution
## first argument is sample size
## second argument is mean
## third argument is SD
C <- rnorm(n,0,1)

# propensity model
## theta is a 2-dimensional vector (list) of parameters for the exposure model
## theta[1] is the intercept and theta[2] is the log-OR for the confounder-exposure relation
## pi is the probability that the exposure is 1
theta <- c(0,log(2))
pi <- expit(theta[1]+theta[1]*C)

# simulate exposure from binomial distribution
## second argument is number of trials
## third argument is probability that A = 1
A <- rbinom(n,1,pi)

# outcome model
```

```
  ## beta is a 3-dimensional vector (list) of parameters for the outcome model
  ## beta[1] is the intercept, beta[2] is the exp(OR) for the exposure-outcome relation
  ## beta[3] is the log-OR for the confounder-outcome relation
  beta <- c(-2.75,log(2),log(2))
  mu <- expit(beta[1] + beta[2]*A + beta[3]*C)
  Y <- rbinom(n,1,mu)
```

Our result of interest from this simulation study will be how collapsibility is affected by the prevalence of the outcome, and whether a $\lessapprox 10\%$ outcome prevalence is sufficient to render the conditional and marginal OR approximately equal. The prevalence of Y can be changed by varying the intercept parameter in the outcome model above (`beta[1]`). To answer our question, we will need to store the prevalence of Y in a variable:

```
  # glm.res0 is the outcome's prevalence
  ## we store this to ouput it from the function
  glm.res0 <- mean(Y)

  print(glm.res0)
```

```
## [1] 0.102
```

So our outcome prevalence is 10.2%. Our dataset consists of three variables, and the first and last three entries are:

```
head(data.frame(Y,A,C=round(C,2)),3)
```

```
##   Y A     C
## 1 0 0 -0.56
## 2 1 1 -0.23
## 3 0 0  1.56
```

```
tail(data.frame(Y,A,C=round(C,2)),3)
```

```
##     Y A    C
```

```
## 498 0 0 0.16
## 499 0 1 0.36
## 500 0 0 0.55
```

We can now estimate the conditionally and marginally adjusted odds ratios in this sample of 500 observations. We can fit a conditionally adjusted model using the following code:

```
# estimate the true exposure odds ratio using a conditonally adjusted logit model
## NB: conditionally adjusted logit model is not the same as "conditional logistic regression"
## glm.res1 is the conditional log-odds ratio

m1 <- glm(Y~A+C,family=binomial(link="logit"))
glm.res1 <- m1$coefficients[2]
```

We can fit a marginally adjusted approach using the following code:

```
# estimate the true exposure odds ratio using a marginally adjusted logit model
## compute the average predicted probabilities under A = 1 and then A = 0
muhat1 <- mean(predict(m1,newdata=data.frame(A=1,C),type="response"))
muhat0 <- mean(predict(m1,newdata=data.frame(A=0,C),type="response"))

## compute the odds from these average probabilities
odds1 <- muhat1/(1-muhat1)
odds0 <- muhat0/(1-muhat0)

## glm.res2 is the marginal log-odds ratio
glm.res2 <- log(odds1/odds0)
```

Summarizing our results, we found that for an outcome prevalence of 10.2%, the conditionally adjusted OR was 2.41, and the marginally adjusted OR was 2.29. Not a huge difference. Keep in mind, this simulation study is extremely limited. We only used a single sample of 500 observations. Thus, the pattern in these results can be explained entirely by random noise. In typical simulation studies, one would repeat the above process 1,000, 5,000 or even 10,000 times, and take the averages of each result.

Let's fix this briefly to see if it changes anything:

```
expit<-function(a){1/(1+exp(-a))}

set.seed(123)

collapsibility_function <- function(index, intercept){
    n=500

    C <- rnorm(n,0,1)

    theta <- c(0,log(2))
    pi <- expit(theta[1]+theta[1]*C)

    A <- rbinom(n,1,pi)

    beta <- c(intercept,log(2),log(2))
    mu <- expit(beta[1] + beta[2]*A + beta[3]*C)
    Y <- rbinom(n,1,mu)

    glm.res0 <- mean(Y)

    m1 <- glm(Y~A+C,family=binomial(link="logit"))
    glm.res1 <- m1$coefficients[2]

    muhat1 <- mean(predict(m1,newdata=data.frame(A=1,C),type="response"))
    muhat0 <- mean(predict(m1,newdata=data.frame(A=0,C),type="response"))

    ## compute the odds from these average probabilities
    odds1 <- muhat1/(1-muhat1)
    odds0 <- muhat0/(1-muhat0)

    ## glm.res2 is the marginal log-odds ratio
    glm.res2 <- log(odds1/odds0)
```

```
    res <- data.frame(prevalenceY = glm.res0,
                      conditionalOR = glm.res1,
                      marginalOR = glm.res2)

    return(res)
}

sim_res <- lapply(1:2000, function(x) collapsibility_function(index = x, intercept = -2.75))

sim_res <- do.call(rbind, sim_res)

head(sim_res)
```

```
##    prevalenceY conditionalOR marginalOR
## A        0.102     0.8788807  0.8304875
## A1       0.112     0.6993057  0.6719350
## A2       0.110     0.4518576  0.4313098
## A3       0.094     0.7279202  0.7009233
## A4       0.080     0.9304425  0.8760232
## A5       0.098     0.9770572  0.9490961
```

```
mean(sim_res$conditionalOR)
```

```
## [1] 0.7177079
```

```
mean(sim_res$marginalOR)
```

```
## [1] 0.6844239
```

Once again, not a major difference. The next step would be to evaluate how the prevalence of the outcome affects the difference in the conditionally and marginally adjusted odds ratios, but we'll leave that for later if time permits.

2 Defining your Data Generating Mechanism Using Directed Acyclic Graphs

In designing a simulation study, it is important to understand how to generate random variables so that the relevant causal and statistical associations between them hold as expected. An important tool in accomplishing this goal is the use of causal diagrams, or directed acyclic graphs. Figures 2 is a directed acyclic graph, which must generally be drawn using rules that govern a DAG, sometimes referred to as the Markov properties of the model.

A directed acyclic graph is a graphical model with **three key properties**:

1. All arrows (edges) are directed from one variable (node) to another.
2. There are no cycles/loops in the diagram.
3. All common causes are included in the DAG.

If any of these properties is not met in a particular graphical model, it is not a DAG. There are several concepts and techniques relevant to the use of DAGs that we cannot cover here. However, for simulation studies, the most important concept related to DAGs is the concept of variable exogeneity and / or endogeneity.

With respect to a particular DAG, a variable is exogenous if it has no arrows pointing into it. On the other hand, the most endogenous variable in a DAG is the one that has the most arrows pointing into it. If we re-visit Figure 2, we can note the following:

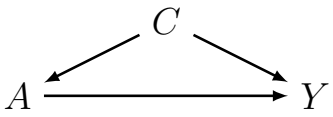

Figure 3: Simple confounding triangle, with exposure A, confounder C, and outcome Y.

- The variable C is most exogenous to this system
- The variable A is second most exogenous
- The variable Y is the most endogenous

Note how this aligns with how we simulated our data:

```r
collapsibility_function <- function(index, intercept) {

    n = 500

    C <- rnorm(n, 0, 1)  ## FIRST!!!

    theta <- c(0, log(2))
    pi <- expit(theta[1] + theta[1] * C)

    A <- rbinom(n, 1, pi)  ## SECOND!!

    beta <- c(intercept, log(2), log(2))
    mu <- expit(beta[1] + beta[2] * A + beta[3] * C)
    Y <- rbinom(n, 1, mu)  ## THIRD!

    ...

}
```

This highlights an important point about how we can construct a data-generating mechanism of interest. If we start with a DAG, we can use each variables relative exogeneity to determine how we can simulate a dataset of interest. As a more complicated example, consider the following mediation diagram:

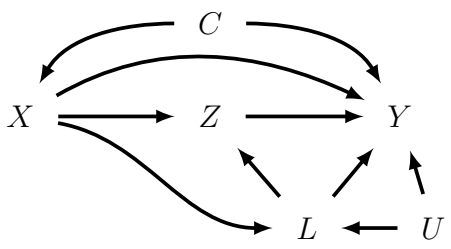

Figure 4: Complex mediation diagram with unmeasured confounder U, baseline confounders C, mediator-outcome confounder affected by the exposure L, mediator Z, exposure X, and outcome Y.

This DAG might be used to motivate a simulation study on the properties of different methods to quantify mediation effects (e.g., direct and indirect

effects). Using the concept of relative exogeneity/endogeneity. We first have to identify the most exogenous variables, and work our way down to the most endogenous variables. We can do this by counting the number of arrows that goes into each variable:

Variable	Arrow Number
U	0
C	0
X	1
L	2
Z	2
Y	5

In this table, there are two sets of variables that share a tie. The first are totally exogenous, and so it doesn't really matter whether we simulate one first or the other. The second set that share a tie are the variables L and Z, each with two arrows. However, in this case, one of the arrows going into Z comes from L. This means we need to simulate L before we simulate Z.

This suggests that we need to simulate our variables in the following order:

$$U, C, X, L, Z, Y$$

We can do this using the following code:

```
expit <- function(x) {
    1/(1 + exp(-x))
}

set.seed(123)

n = 500

U <- rnorm(n)

C <- rnorm(n)
```

```r
X <- rbinom(n, size = 1, p = expit(-1 + log(2) * C))

L <- rnorm(n, mean = 0 + 1.5 * X + 1.5 * U)

Z <- rbinom(n, size = 1, expit(-1.5 + log(2) * X + log(2) * L))

Y <- rnorm(n, mean = 10 + 5 * X + 5 * C + 5 * Z + 4 * L + 4 *
    U, sd = 5)

med_data <- data.frame(Y, Z, L, X, C)

head(med_data)
```

```
##           Y Z         L X           C
## 1  3.034566 0 0.2109878 0 -0.60189285
## 2  6.406043 0 0.2776392 0 -0.99369859
## 3 29.747544 0 2.7716829 0  1.02678506
## 4 22.100866 0 0.4918470 0  0.75106130
## 5  9.783017 0 1.4852549 0 -1.50916654
## 6 19.589532 0 1.5703376 0 -0.09514745
```

This flexible procedure is a useful tool in generating data from potentially complex data structures. Note, however, that there is a whole lot in addition to the ordering of variables that has to be decided to successfully simulate data. For instance, we simulated the outcome Y based on a normal distribution with a constant variance. We could have used a binomial or Poisson distribution. We did not include any interactions between any of these variables in our simulation. This should be determined on the basis of the question of interest motivating the simulation. Finally, and perhaps most importantly, we selected coefficients for our outcome model in such a way that isolates the independent effect of each variable, without considering the overall effect that we might be interested in. We'll discuss this issue in the next sections.

3 What is Your Estimand?

In a given study, the estimand is the target parameter we seek to quantify with the study data. In a simulation setting, the estimand can be identified as the true underlying relationship in the data that we are primarily interested in quantifying. For example, in the above Figure 2, there are two possible estimands that we can adopt as our target. The first is the conditionally adjusted odds-ratio, which in the context of our simulated data is 2. Here is that code again:

```
# outcome model beta is a 3-dimensional vector (list) of
# parameters for the outcome model beta[1] is the
# intercept, beta[2] is the exp(OR) for the
# exposure-outcome relation beta[3] is the log-OR for the
# confounder-outcome relation
beta <- c(-2.75, log(2), log(2))
mu <- expit(beta[1] + beta[2] * A + beta[3] * C)
Y <- rbinom(n, 1, mu)
```

The second estimand we can pick in this setting is the marginally adjusted odds-ratio, but the actual value of this estimand is more difficult to determine. In fact, it will require integration to solve for:

$$\mu(A = a) = \int_C \text{expit}\{\beta_0 + \beta_1 a + \beta_2 C\} dC$$

which we can then use to compute the marginally adjusted odds ratio:

$$\frac{\mu(A = 1)}{1 - \mu(A = 1)} \Big/ \frac{\mu(A = 0)}{1 - \mu(A = 0)}$$

The problem is that, even in this simple single C setting, computing this integral is challenging because of the expit function and the fact that C is normally distributed. In this case, I never attempt to solve for the true value of the marginally adjuted odds ratio (or marginally adjusted effect, more generally). Instead, I rely on the Oracle method, which will be described below.

4 No, really, What is Your Estimand?

As we will soon see, we need the true value of the estimand in any situation we are interested in quantifying bias, mean squared error, or most measures of estimator performance we usually want from a simulation study. But the challenge in solving for the true parameter value can be real. Take, for example, Figure 4 and suppose we are interesed in the effect defined as a contrast of the outcome that would be observed if everyone in the population were exposed versus unexposed. How can we determine this from the code we used to generate our data? This effect is actually represented in the DAG (Figure 4) by the combined arrows emanating from X into Y. This includes:

Path
$X \rightarrow Y$
$X \rightarrow Z \rightarrow Y$
$X \rightarrow L \rightarrow Y$
$X \rightarrow L \rightarrow Z \rightarrow Y$

We actually know the magnitude of each arrow represented in this Table. The tough question is, how do we combine them?

5 The True Value and Monte Carlo Integration

Fortunately, there is a general technique that we can use to determine the actual numerical value of interest, without having to solve for complex integral equations or determine how to combine multiple effect magnitudes of interest into a single estimand value of interest. To illustrate this technique, let's start with the simpler data generating mechanism evaluating the difference between the conditionally and marginally adjusted odds ratio. Here's the data generating mechanism we used, with a few key differences. First, we put an argument for the exposure in the function. Second, we generate only the marginally adjusted odds under a specific exposure value. Third, we increase the sample size to as large as we can tolerate:

```r
expit <- function(a) {
    1/(1 + exp(-a))
}

set.seed(123)

collapsibility_function <- function(intercept, exposure) {
    n = 5e+06  # five million observations

    C <- rnorm(n, 0, 1)

    theta <- c(0, log(2))
    pi <- expit(theta[1] + theta[1] * C)

    A <- exposure  # set the exposure to a specific value

    beta <- c(intercept, log(2), log(2))
    mu <- expit(beta[1] + beta[2] * A + beta[3] * C)
    Y <- rbinom(n, 1, mu)

    mu_ <- mean(Y)  # output the mean of Y under the specific exposure value

    ## compute the odds from these average probabilities
    odds <- mu_/(1 - mu_)

    ## output the marginally adjusted odds
    return(odds)
}
```

In this modified function, we can now compute the odds from the data generating mechanism under a condition where everyone is exposed:

```r
odds1 <- collapsibility_function(intercept = -2.75, exposure = 1)

odds1
```

```
## [1] 0.151236
```

We can do the same thing where everyone is unexposed:

```
odds0 <- collapsibility_function(intercept = -2.75, exposure = 0)

odds0
```

```
## [1] 0.07839244
```

We can now take the ratio of these two odds to quantify the true value using Monte Carlo integration:

```
true_ORm <- odds1/odds0

true_ORm
```

```
## [1] 1.929217
```

When implementing this procedure, it's important to NOT change change anything else except the exposure status and the sample size in the function used to conduct the simulation. If something else is changed, you may end up quantifying an estimand that does not correspond to the effect of interest that you're after.

References

Sander Greenland. Collapsibility. In Mitchell H. Gail and Jacques Bénichou, editors, *Encyclopedia of Epidemiologic Methods*. John Wiley & Sons, Ltd, 2005.

Sander Greenland and Judea Pearl. Adjustments and their consequences—collapsibility analysis using graphical models. *International Statistical Review*, 79(3):401–426, 2011. ISSN 1751-5823. DOI: 10.1111/j.1751-5823.2011.00158.x. URL http://dx.doi.org/10.1111/j.1751-5823.2011.00158.x.

Sander Greenland, James M. Robins, and Judea Pearl. Confounding and collapsibility in causal inference. *Stat Sci*, 14(1):29–46, 1999.

Miguel A Hernán, David Clayton, and Niels Keiding. The simpson's paradox unraveled. *International Journal of Epidemiology*, 40(3):780–785, 06 2011. DOI: 10.1093/ije/dyr041. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3147074/.

Tim P. Morris, Ian R. White, and Michael J. Crowther. Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11):2074–2102, 2019.

Menglan Pang, Jay S Kaufman, and Robert W Platt. Studying noncollapsibility of the odds ratio with marginal structural and logistic regression models. *Stat Methods Med Res*, Oct 2013a. DOI: 10.1177/0962280213505804.

Menglan Pang, Jay S Kaufman, and Robert W Platt. Mixing of confounding and non-collapsibility: a notable deficiency of the odds ratio. *Am J Cardiol*, 111 (2):302–303, Jan 2013b. DOI: 10.1016/j.amjcard.2012.09.002.