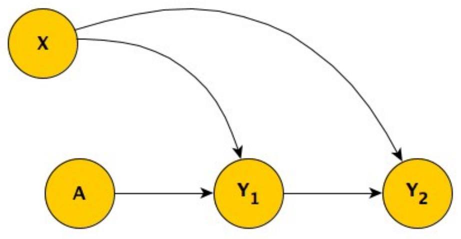
A little background for context

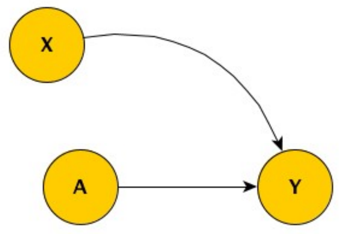
The actual RCT is considerably more complicated than I am describing here, but this is the general idea. Individuals are randomized to one of two study arms \(A\), where \(A_i=1\) if patient \(i\) is in the treatment arm, and \(A_i = 0\) if the the patient is in the control arm. We measure the outcome \(Y\) at two time points, so we have \(Y_1\) and \(Y_2\); our primary interest, however, is \(Y_2\). We measure a key covariate \(X\) that influences both \(Y_1\) and \(Y_2\). This is the true underlying DAG:



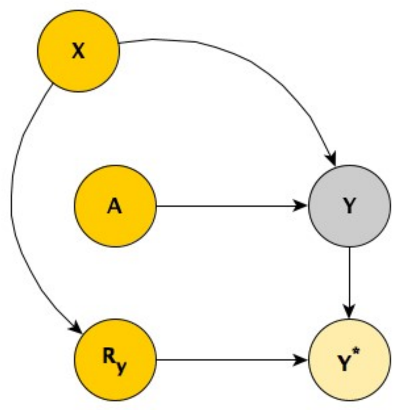
The challenge is that, for some patients, the second measurement \(Y_2\) is missing, and we believe that \(Y_1\) is a good predictor of the missingness pattern. But before getting into this (which is Case #3), I'll start with a simpler scenario.

Case #1

In the first scenario, there is only a single outcome measurement \(Y\), and we have measured \(X\). The simplified DAG looks like this:



Unfortunately, we've only been able to collect the outcome measurement \(Y\) for a subset of the sample, so that the observed \(Y^*\) includes missing values for some subjects. The missing data mechanism is MAR, because the level of the observed baseline covariate \(X\) determines the probability of observing \(Y\). The indicator \(R_y = 1\) when we do observe \(Y\) and \(R_y = 0\) when we do not.



I'll go ahead and simulate data based on this first DAG. In case you'd like to replicate, here are the libraries necessary for the simulations:

```
library(simstudy)
library(ggplot2)
library(broom)
library(data.table)
library(mice)
```

The data definitions establish the relationship between \(A\), \(X\) and \(Y\) (the treatment effect of \(A\) on \(Y\) is 2.5) as well as create a missingness mechanism for \(Y\) that is a function \(X\); subjects with higher values of \(X\) are more likely to have missing outcome measurements.

```
def1 <- defData(varname = "x", formula=0.5, dist = "binary")
def2 <- defDataAdd(varname = "y", formula = "5 + 5*x + 2.5*a", variance
= 2)
defm <- defMiss(varname = "y", formula = "-3.5 + 2.3*x", logit.link =
TRUE)
```

To generate the observed data with missing data, we first generate a complete data set (based on the data definitions), and then we generate a missing data matrix, which finally gives us the observed data set which includes \(\text{NA}\)'s for about 13% of the \(Y\)'s.

```
set.seed(17236)

dd <- genData(500, def1)
dd <- trtAssign(dd, grpName = "a")
dd <- addColumns(def2, dd)

ddmiss <- genMiss(dd, defm, id = "id")
ddobs <- genObs(dd, ddmiss, id = "id")

ddobs
##       id x a    y
##   1:   1 0 0  6.1
##   2:   2 1 0  9.2
##   3:   3 1 1 11.6
##   4:   4 0 0  4.5
##   5:   5 1 1   NA
##  ---
## 496: 496 0 0  5.8
## 497: 497 0 1  7.3
## 498: 498 0 1  6.9
## 499: 499 1 1 11.1
## 500: 500 1 0 10.0
```

Using the full data set `dd` (without any missing data), we can get a point estimate of the treatment effect \(\delta\) merely by calculating

\[\hat{\delta} = \bar{Y}_{a=1} - \bar{Y}_{a=0}\]

```
dd[, .(avg = mean(y)), keyby = a][ , avg - shift(avg)][2]
## [1] 2.5
```

There is no reason to believe that the observed data means are the same as the complete data set means. That is, it is not likely that $\bar{Y^*}_{a=1}$ = $\bar{Y}_{a=1}$ or $\bar{Y^*}_{a=0}$ = $\bar{Y}_{a=0}$. Observations with higher values of X (and thus higher values of Y) are more likely to have missing Y's, so the average observed values in both treatment groups should be lower. This seems to be the case here:

```
dd[, .(avg = mean(y)), keyby = a]
##    a  avg
## 1: 0  7.5
## 2: 1 10.0
ddobs[, (avg = mean(y, na.rm = TRUE)), keyby = a]
##    a  V1
## 1: 0 7.2
## 2: 1 9.7
```

In the real world, we can only estimate the treatment effect δ^* with the data that we have: $$\hat{\delta}^* = \bar{Y}_{a=1}^* - \bar{Y}_{a=0}^*$$

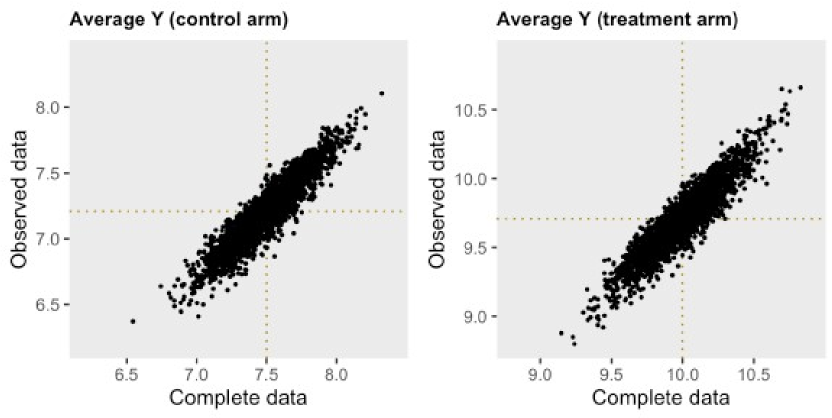
It looks like, in this case at least, the bias in estimates of the means are in the same direction, so that the estimate of the treatment effect based on the *difference* of means in the observed data is unbiased:

```
ddobs[!is.na(y), .(avg = mean(y)), keyby = a][ , avg - shift(avg)][2]
## [1] 2.5
```

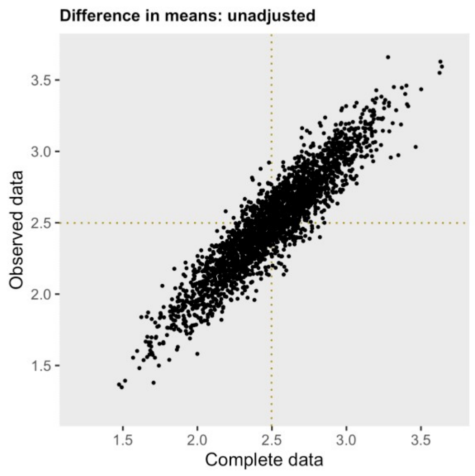
If this is the case more generally for data sets generated using this mechanism, we may not need to worry at all about the missing data mechanism; even though we know it is MAR, we might be able to treat it as MCAR, and just use the observed measurements only, without any adjustment or imputation.

Simulating 2500 data sets using steps outlined above provides insight into the nature of the bias. (I've provided generic code for generating repeated data sets in the addendum.) The estimates based on the complete data set are shown on the x axis, and the observed data estimates are on the y axis. The dotted lines show the average of the estimates for the complete and observed data sets, respectively.

For both treatment arms, the average estimate from the complete data sets is centered around the true value (used in the data generation process). As expected (since higher values of Y are likely to be missing), the average estimate for each arm is biased downwards when we do not take into consideration the missingness.

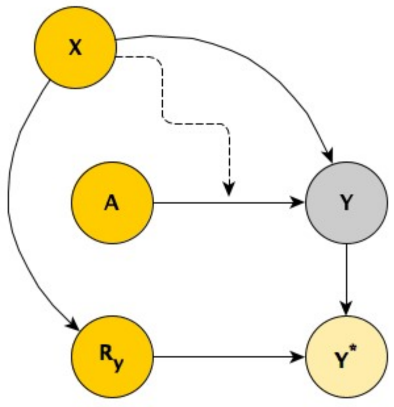


However, the bias is removed when we consider the treatment effect, which is our primary interest. In this (perhaps overly) simplistic scenario, there is no price to pay when ignoring the missing data. Both estimates are centered around 2.5, the true value.



Case #2

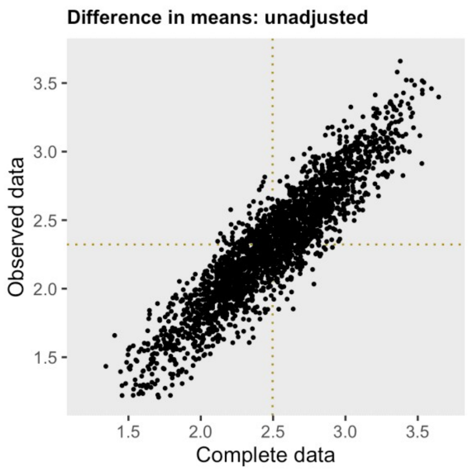
The second example differs from the first only in one respect: the size of the intervention effect depends on the baseline covariate \(X\) (the line drawn from \(X\) to the arrow connecting \(A\) and \(Y\) represents this effect modification).



In the example, \(\delta_0 = 1\) for the sub-population with \(X = 0\), and \(\delta_1 = 4\) for the sub-population with \(X = 1\). If the population were evenly distributed between \(X=0\) and \(X=1\), then we would observe an overall effect \(\delta = 2.5\).

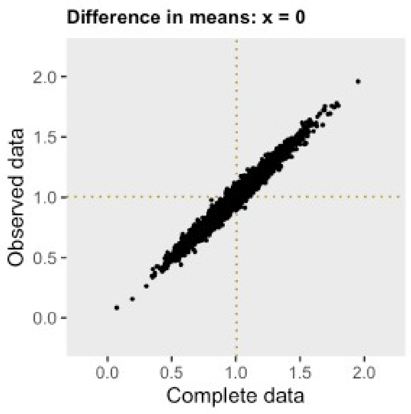
```
d1 <- defData(varname = "x", formula=0.5, dist = "binary")
d2 <- defDataAdd(varname = "y", formula = "6 + 1*a + 2*x + 3*a*x",
variance = 2)
dm <- defMiss(varname = "y", formula = "-3.5 + 2.3*x", logit.link =
TRUE)
```

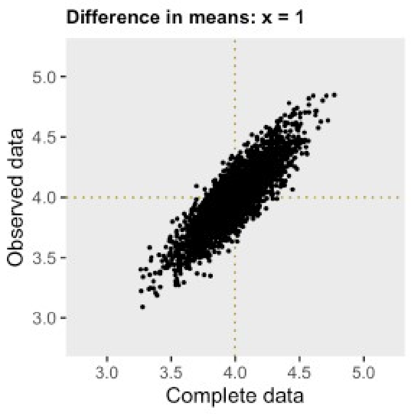
But this time around, if we go ahead and naïvely estimate \(\delta^* = \bar{Y}_{a=1}^* - \bar{Y}_{a=0}^*\), the estimate will be biased.



The reason for this bias is that the mix of \(X\) in the observed sample is different from the complete sample (and population); since \(X\) influences the effect size this change impacts the overall unadjusted estimate. In the complete data set \(P(X=1) = 0.50\), but in an observed data set with missing values \(P^*(X=1) = 0.44\) (this can be confirmed using the assumptions from the data generation process, but I'll let you do that as an exercise if you'd like.) The population average treatment effect is \(P(X=0) \times 1 + P(X=1) \times 4 = 2.5\). And in the data set with missing data \(P^*(X=0) \times 1 + P^*(X=1) \times 4 = 0.56 \times 1 + 0.44 \times 4 = 2.3\).

We can still estimate the treatment effect if we adjust for \(X\) in a regression model, or just take the difference in means within each level of \(X\). These estimates are unbiased:

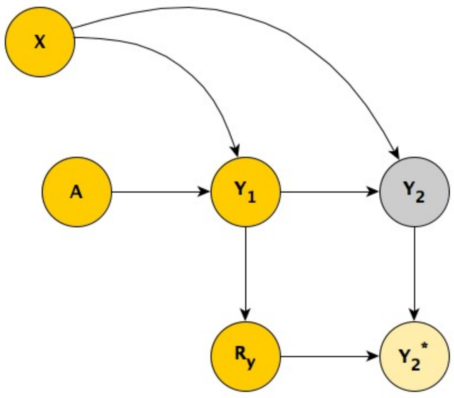




If we want to recover the population average treatment effect, we can reweight the group-level treatment effects by the distribution of \(X\) in complete sample (since \(X\) is fully observed). No imputation is needed.

Case #3

Now we are back to the original motivating scenario. The missing data mechanism is depicted in the next DAG. Those with higher scores in the first period are more likely to have missing values in the second time period, perhaps because they have improved sufficiently and no longer feel like participating in the study.



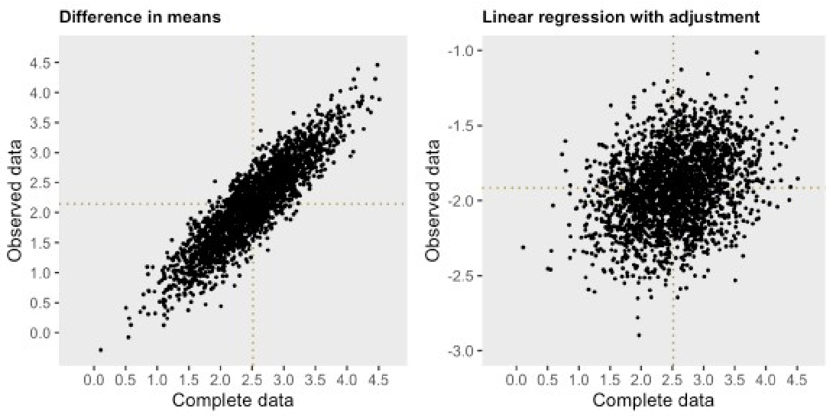
The DAG is implemented with these definitions:

```
def1 <- defData(varname = "x", formula=0.5, dist = "binary")

def2 <- defDataAdd(varname = "y1", formula = "5 + a*2.5 + 5*x",
variance = 2)
def2 <- defDataAdd(def2, "y2", formula = "1 + y1 + 5*x", variance = 2)

defm <- defMiss(varname = "y2", formula = "-4.5 + 0.3*y1", logit.link =
TRUE)
```

In this case, simply comparing the means in the data sets with missing data provides a biased estimate - we can see this on the left; the argument is similar to the one I made in the previous scenario. If we opt to control for \(Y_1\), we introduce all sorts of biases, as \(Y_1\) is a mediator between \(A\) and \(Y_2\), as well as a collider. (I've written about the dangers of controlling for post-intervention variables here and the need to be careful with colliders here.)

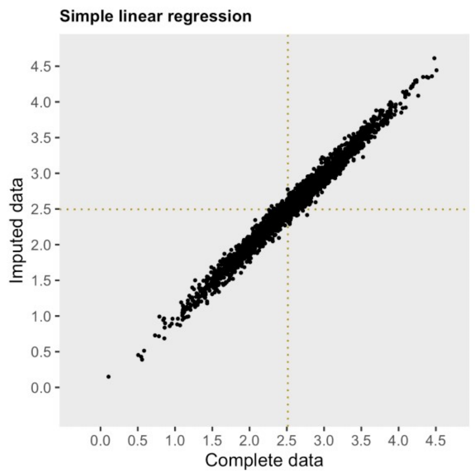


Since neither a simple comparison of means nor an adjusted model will suffice here, our only option is to use multiple imputation, which in `R` can be can be implemented with the package mice. Below, I am showing code that generates 20 imputed data sets, fits models for each of them, and pools the results to provide a single estimate and measure of uncertainty.

```
library(mice)

imp <- mice(ddobs[,-"id"], m=20, maxit=5, print=FALSE)
fit <- with(imp, lm(y2 ~ a))
results <- summary(pool(fit))
```

Multiple imputation has been applied to the same 2500 data sets with missing data that are represented in the biased estimate plots. The plot below shows a pretty strong correlation with the estimates from the full data mode, and both are centered at the true population effect of 2.5.



The takeaway from all this is that while multiple imputation is not always necessary, if you think there are potentially unmeasured confounders or post-intervention measures that are conceivably in the mix, a multiple imputation approach might be wiser than trying to adjust your way out of the problem.

I plan on implementing a Bayesian model that treats the missing data as parameters. If I can get that working, I will share it here, of course.

Addendum

In case you'd like to play around with other scenarios, I'm including the code that will allow you to repeatedly sample data sets. Just provide you our data definitions.

```
s_generate <- function(n) {

  dd <- genData(n, d1)
  dd <- trtAssign(dd, grpName = "a")
  dd <- addColumns(d2, dd)

  dmiss <- genMiss(dd, dm, id = "id")
  dobs <- genObs(dd, dmiss, id = "id")

  return(list(dd, dobs))

}

s_replicate <- function(n) {

  dsets <- s_generate(n)

  diff.complete <- dsets[[1]][, .(avg = mean(y2)), keyby = a][ , avg -
shift(avg)][2]
  diff.obs<- dsets[[2]][!is.na(y2), .(avg = mean(y2)), keyby = a][ , avg
- shift(avg)][2]

  est.complete <- coef(lm(y2 ~ a, data = dsets[[1]]))["a"]
  est.obs <- coef(lm(y2 ~ y1 + a, data = dsets[[2]]))["a"]

  imp <- mice(dsets[[2]][,-"id"], m=20, maxit=5, print=FALSE)
  fit <- with(imp, lm(y2 ~ a))
  pooled.ests <- summary(pool(fit))
  est.impute <- pooled.ests$estimate[2]

  return(data.table(diff.complete, est.complete, diff.obs, est.obs,
est.impute))
}

results <- rbindlist(mclapply(1:2500, function(x) s_replicate(300),
mc.cores = 4))
```