G-Computation

Malcolm Barrett

RStudio, PBC

2021-09-01 (updated: 2022-10-16)

Normal regression estimates associations. But we want **causal** estimates: what would happen if **everyone** in the study were exposed to x vs if **no one** was exposed.

G-Computation/G-Formula

1. **Fit a model for** y ~ x + z **where z is all covariates**
2. **Create a duplicate of your data set for each level of** x
3. **Set the value of x to a single value for each cloned data set (e.g** x = 1 **for one,** x = 0 **for the other)**

G-Computation/G-Formula

1. **Make predictions using the model on the cloned data sets**
2. **Calculate the estimate you want, e.g.** mean(x_1) - mean(x_0)

Advantages of the parametric G-formula

Often more statistically precise than propensity-based methods

Incredibly flexible

Basis of other important causal models, e.g. causal survival analysis and TMLE

Greek Pantheon data (greek_data)

name	l	a	y
Rheia	0	0	0
Kronos	0	0	1
Demeter	0	0	0
Hades	0	0	0
Hestia	0	1	0
Poseidon	0	1	0
Hera	0	1	0
Zeus	0	1	1
Artemis	1	0	1
Apollo	1	0	1

+ 10 more rows

1. Fit a model for y ~ a + l

```
greek_model <- lm(y ~ a + l, data = greek_data)
```

2. Create a duplicate of your data set for each level of a

name	l	a	y
Rheia	0	0	0
Kronos	0	0	1
Demeter	0	0	0
Hades	0	0	0
Hestia	0	1	0
Poseidon	0	1	0
Hera	0	1	0
Zeus	0	1	1
Artemis	1	0	1
Apollo	1	0	1

2. Create a duplicate of your data set for each level of a

name	l	a	y
Rheia	0	0	0
Kronos	0	0	1
Demeter	0	0	0
Hades	0	0	0
Hestia	0	1	0
Poseidon	0	1	0
Hera	0	1	0
Zeus	0	1	1
Artemis	1	0	1
Apollo	1	0	1

name	l	a	y
Rheia	0	0	0
Kronos	0	0	1
Demeter	0	0	0
Hades	0	0	0
Hestia	0	1	0
Poseidon	0	1	0
Hera	0	1	0
Zeus	0	1	1
Artemis	1	0	1
Apollo	1	0	1

3. Set the value of a to a single value for each cloned data set

name	l	a	y
Rheia	0	**0**	0
Kronos	0	**0**	1
Demeter	0	**0**	0
Hades	0	**0**	0
Hestia	0	**0**	0
Poseidon	0	**0**	0
Hera	0	**0**	0
Zeus	0	**0**	1
Artemis	1	**0**	1
Apollo	1	**0**	1

name	l	a	y
Rheia	0	**1**	0
Kronos	0	**1**	1
Demeter	0	**1**	0
Hades	0	**1**	0
Hestia	0	**1**	0
Poseidon	0	**1**	0
Hera	0	**1**	0
Zeus	0	**1**	1
Artemis	1	**1**	1
Apollo	1	**1**	1

3. Set the value of a to a single value for each cloned data set

```r
#  set all participants to have a = 0
untreated_data <- greek_data %>%
  mutate(a = 0)

#  set all participants to have a = 1
treated_data <- greek_data %>%
  mutate(a = 1)
```

4. Make predictions using the model on the cloned data sets

```r
#  predict under the data where everyone is untreated
predicted_untreated <- greek_model %>%
  augment(newdata = untreated_data) %>%
  select(untreated = .fitted)

#  predict under the data where everyone is treated
predicted_treated <- greek_model %>%
  augment(newdata = treated_data) %>%
  select(treated = .fitted)

predictions <- bind_cols(
  predicted_untreated,
  predicted_treated
)
```

5. Calculate the estimate you want

```
predictions %>%
  summarise(
    mean_treated = mean(treated),
    mean_untreated = mean(untreated),
    difference = mean_treated - mean_untreated
  )
```

```
## # A tibble: 1 × 3
##   mean_treated mean_untreated difference
##          <dbl>          <dbl>      <dbl>
## 1          0.5            0.5          0
```

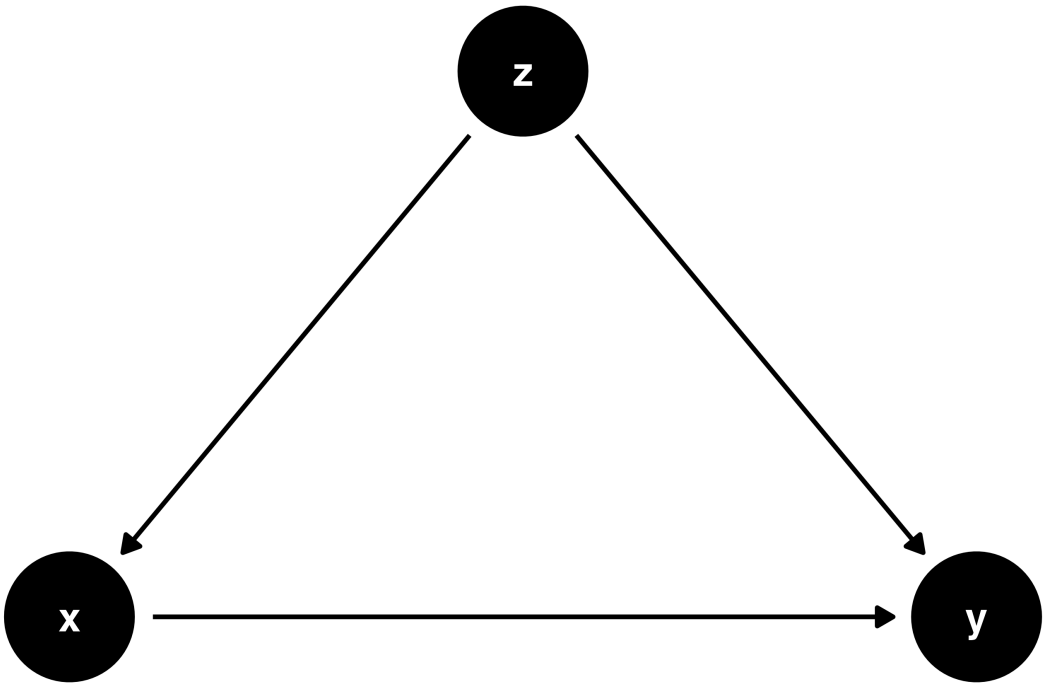
Your Turn

Work through Your Turns 1-3 in 07-g-computation-exercises.Rmd

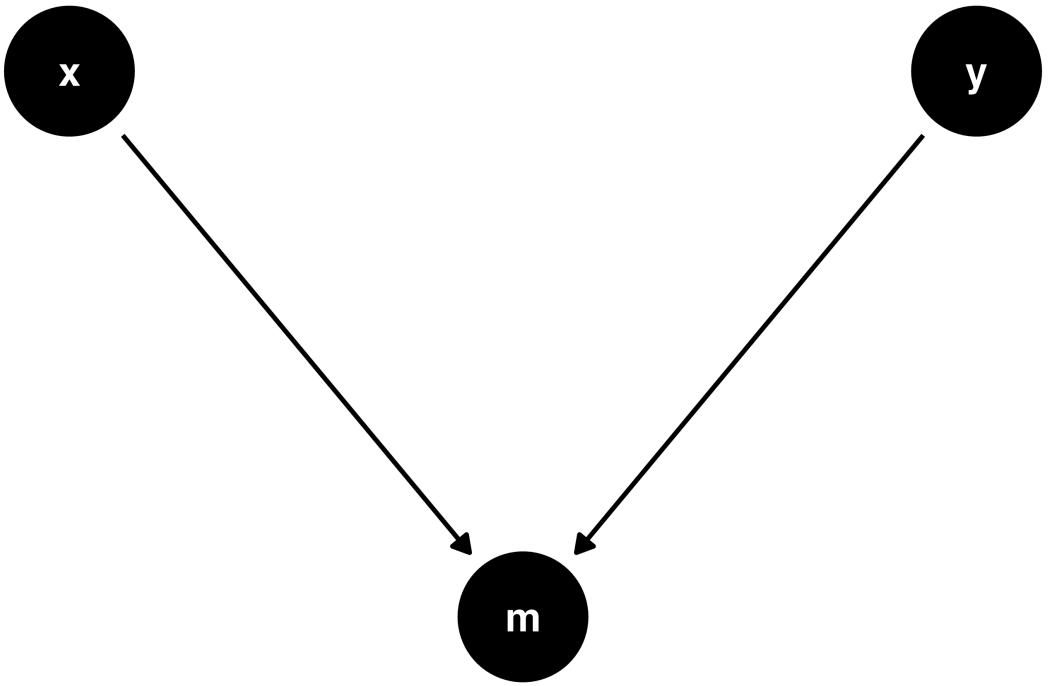
10:00

Detour: Colliders, selection bias, and loss to follow-up

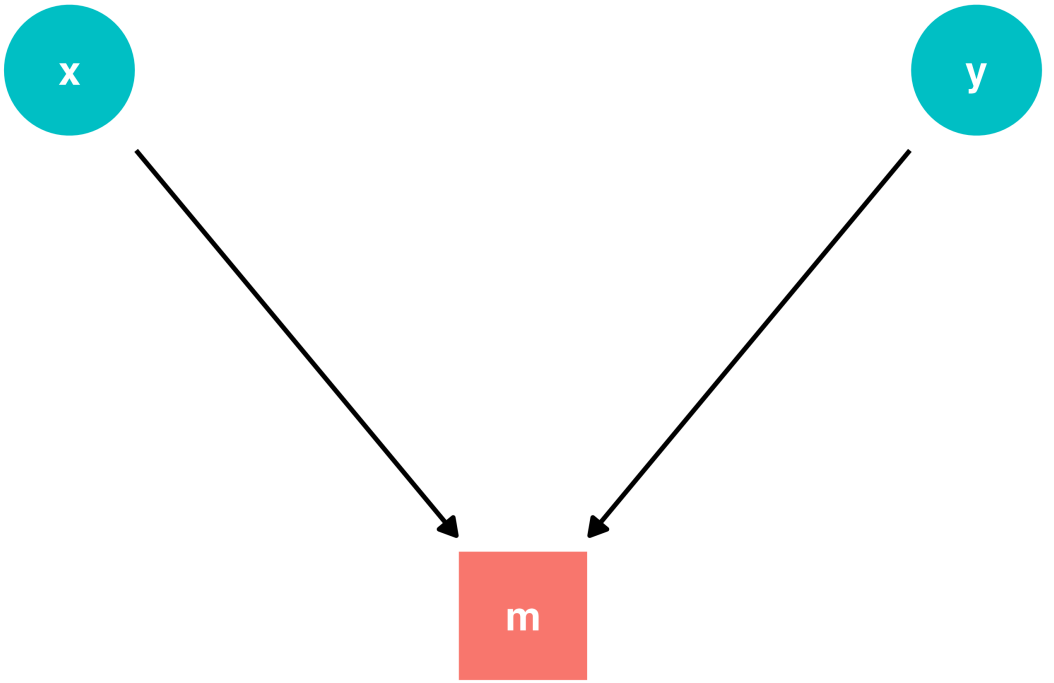
Confounders and chains



Colliders



Colliders



Let's prove it!

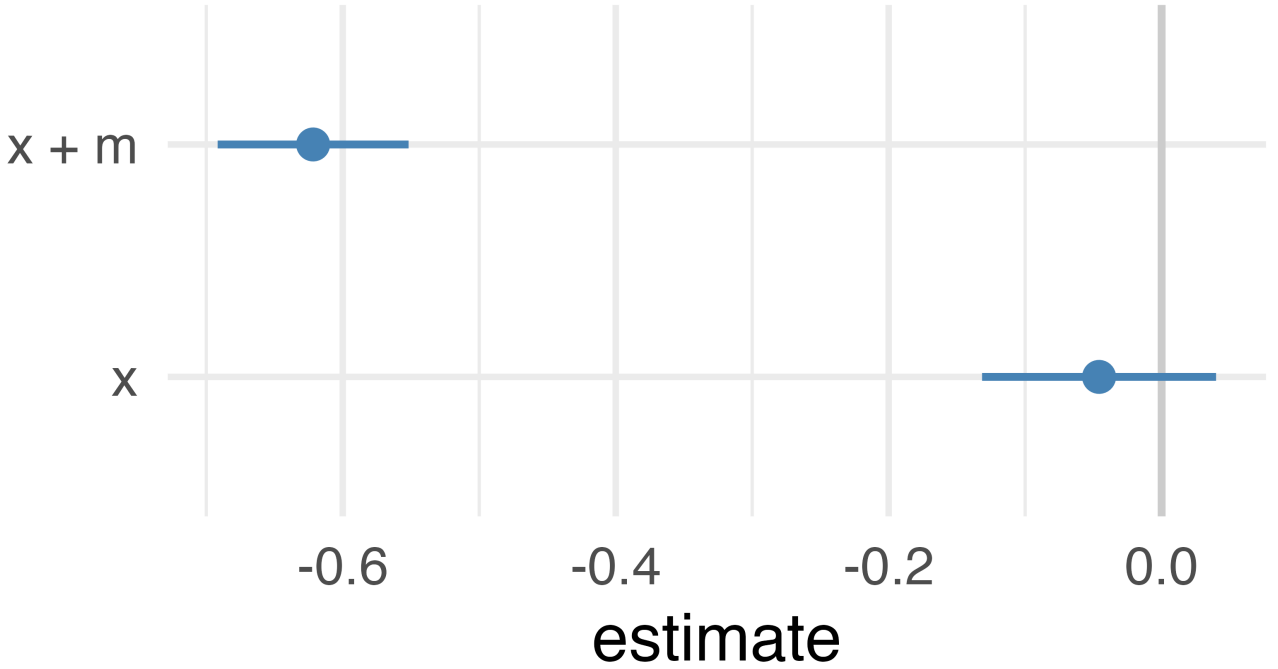
```
set.seed(1234)
collider_data <- collider_triangle() |>
  simulate_data(-.6)
```

Let's prove it!

```
collider_data
```

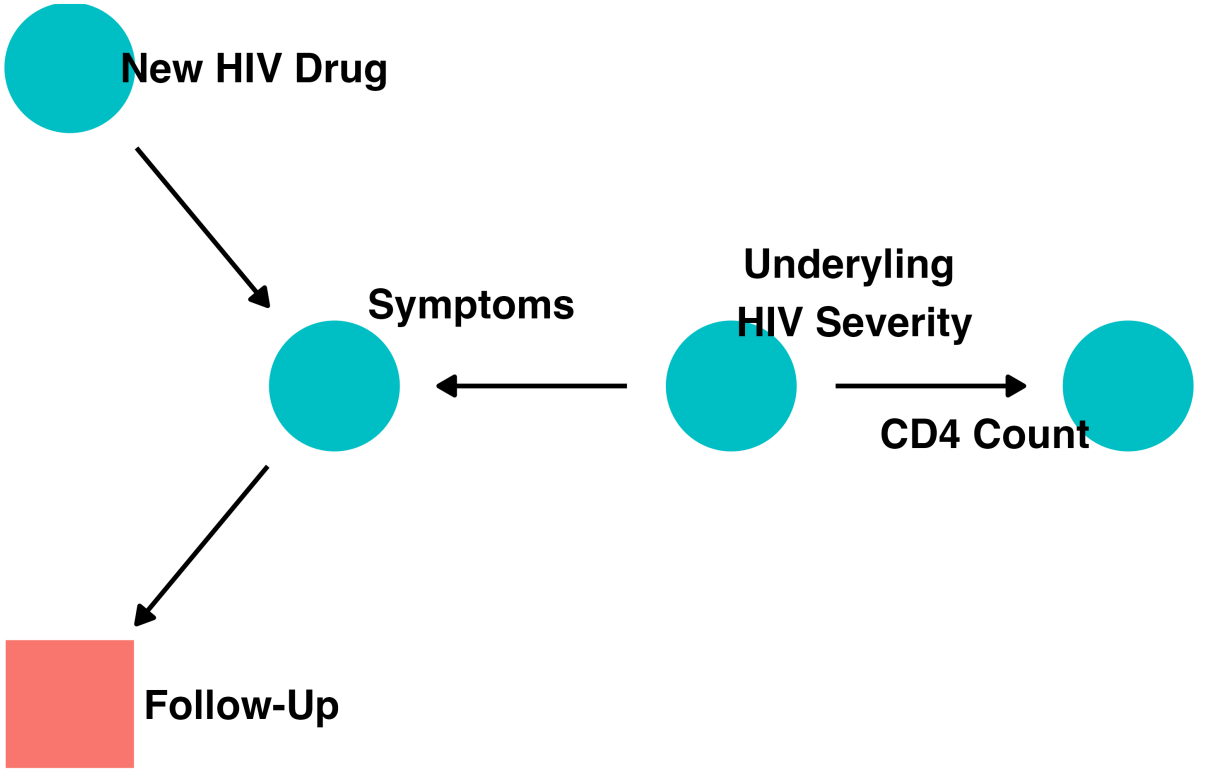
```
## # A tibble: 500 × 3
##           m      x          y
##       <dbl>  <dbl>      <dbl>
##  1 -0.457  -0.410   1.28
##  2  0.281   1.79   -0.550
##  3  0.0835  1.31   -0.169
##  4  0.640   1.06   -1.40
##  5 -1.30    0.435   1.16
##  6 -0.569   0.630  -0.000667
##  7 -0.793   1.50   -1.10
##  8 -0.482   0.748  -0.411
##  9 -0.706   1.03   -0.381
## 10  1.42   -0.841  -0.420
## # … with 490 more rows
```

Let's prove it!



correct effect size: 0

Loss to follow-up



Adjusting for selection bias

1. Fit a probability of censoring model, e.g. `glm(censoring ~ predictors, family = binomial())`
2. Create weights using inverse probability strategy
3. Use weights in your causal model

We won't do it here, but you can include many types of weights in a given model. Just take their product, e.g. **multiply inverse propensity of treatment weights by inverse propensity of censoring weights**.

Your Turn

Work through Your Turns 4-6 in 07-g-computation-exercises.Rmd

10:00