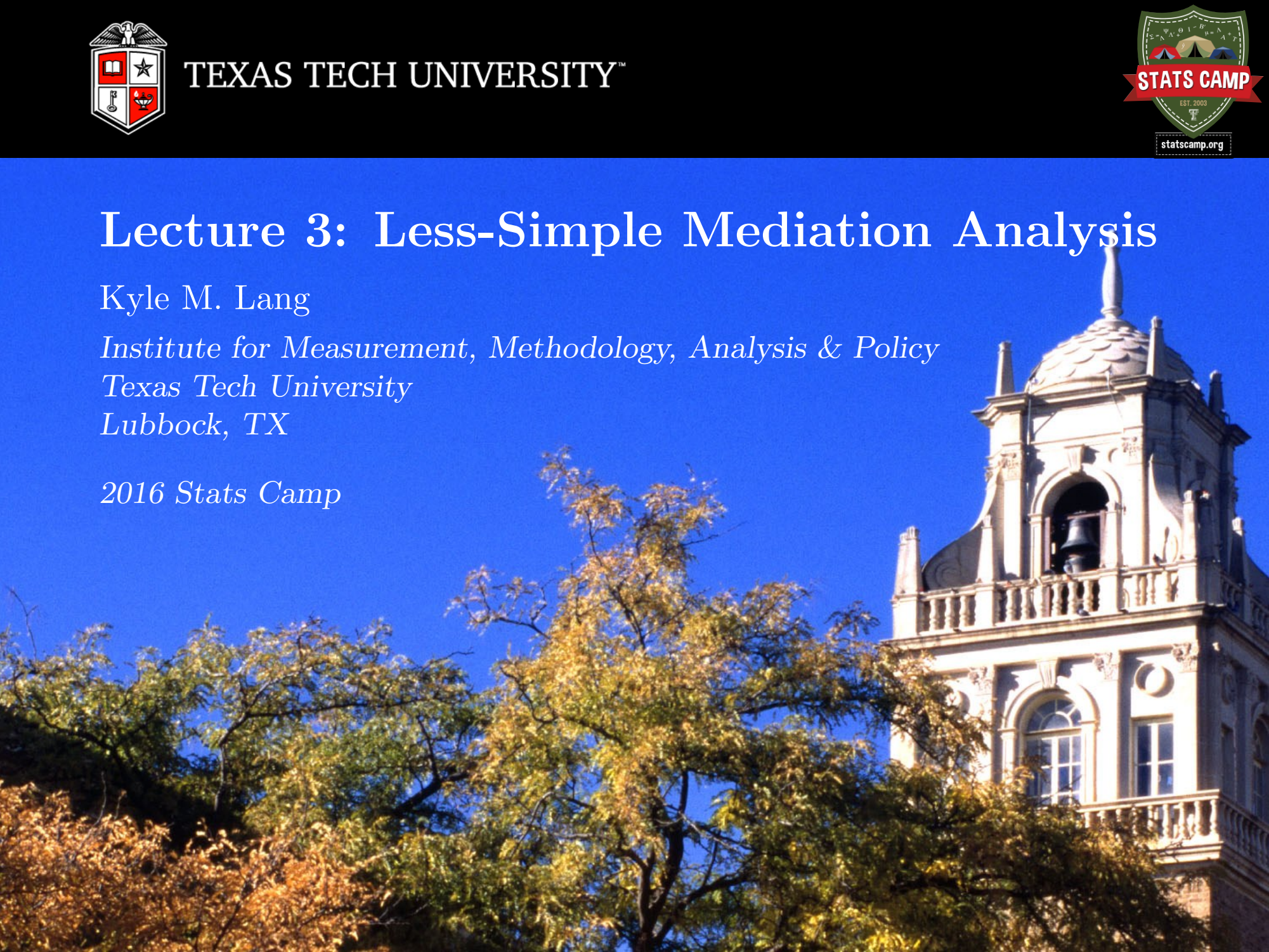


Lecture 3: Less-Simple Mediation Analysis

Kyle M. Lang

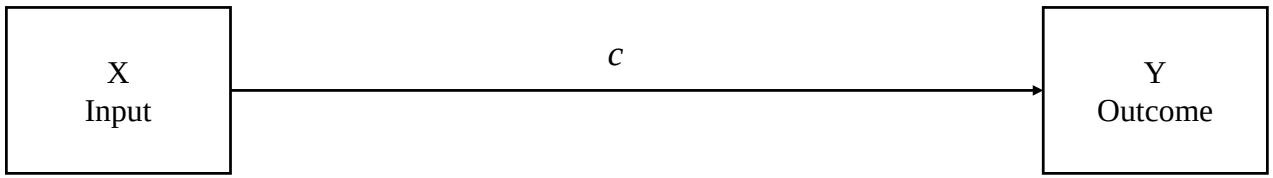
Institute for Measurement, Methodology, Analysis & Policy
Texas Tech University
Lubbock, TX

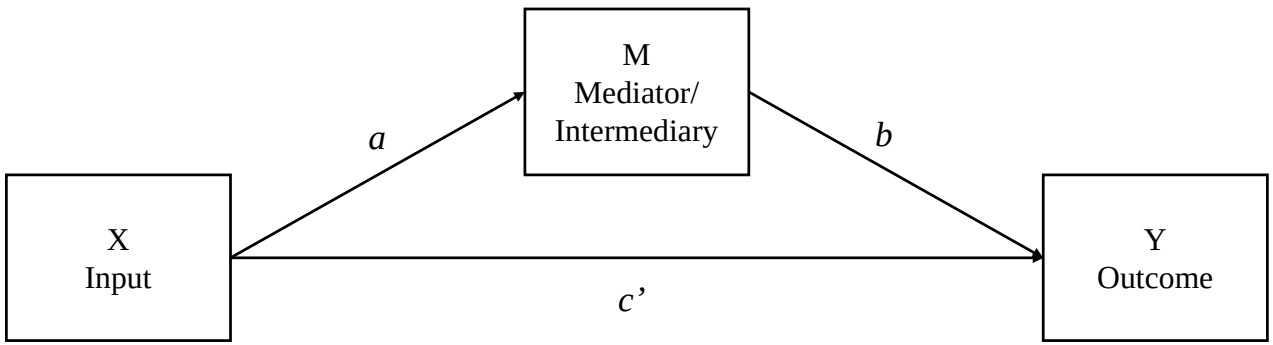
2016 Stats Camp

Outline

- Show how to estimate indirect effects with path diagrams
- Discuss the use of *bootstrapping* for testing indirect effects
- Discuss the Monte Carlo simulation method for testing indirect effects

Recall our Basic Path Diagram





The Basic OLS Equations

To get all the pieces of the preceding diagram using OLS regression, we're forced fit three equations.

$$Y = i_1 + cX + e_1 \tag{1}$$
$$Y = i_2 + c'X + bM + e_2 \tag{2}$$
$$M = i_3 + aX + e_3 \tag{3}$$

- Equation 1 gives us the total effect (c).
- Equation 2 gives us the direct effect (c') and the partialled effect of the mediator on the outcome (b).
- Equation 3 gives us the effect of the input on the outcome (a).

Two Measures of Indirect Effect

Recall the two definitions of an indirect effect:

$$IE_{diff} = c - c' \tag{4}$$

$$IE_{prod} = a \cdot b \tag{5}$$

It pays to remember a few key points:

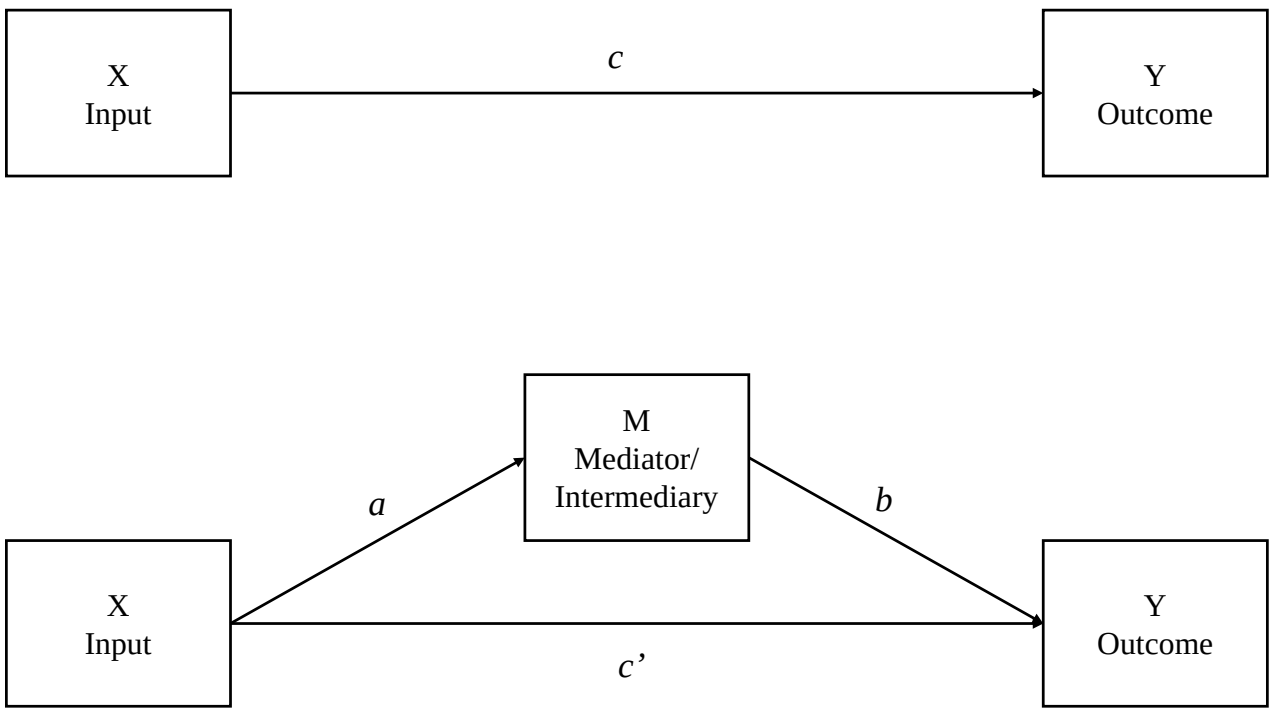
- IE_{diff} and IE_{prod} are equivalent in simple mediation.
- IE_{diff} is only an indirect indication of the parameter we're truly interested in IE_{prod}.
- A significant indirect effect can exist without a significant total effect.
- If we hypothesize a significant indirect effect, then we don't care about the total effect.

These points imply something interesting:

- We don't need to estimate c!

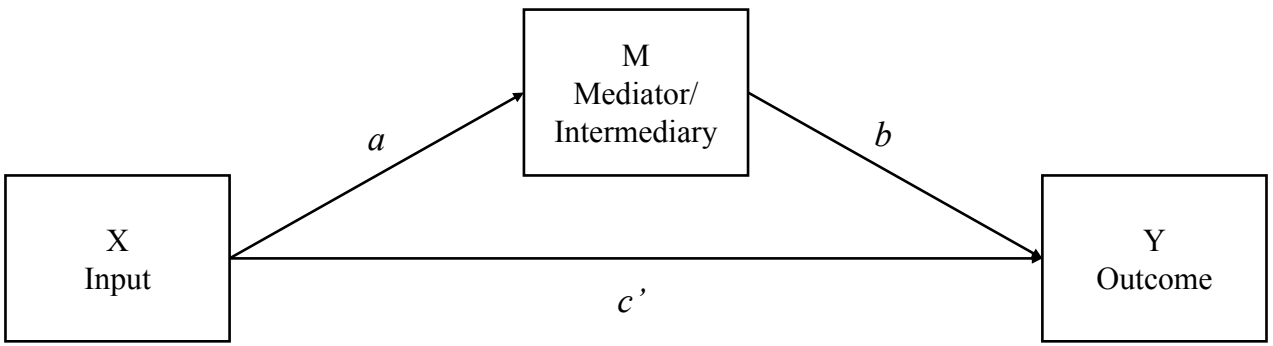
Simplifying our Path Diagram

QUESTION: If we don't care about estimating c, how can we simplify this diagram?



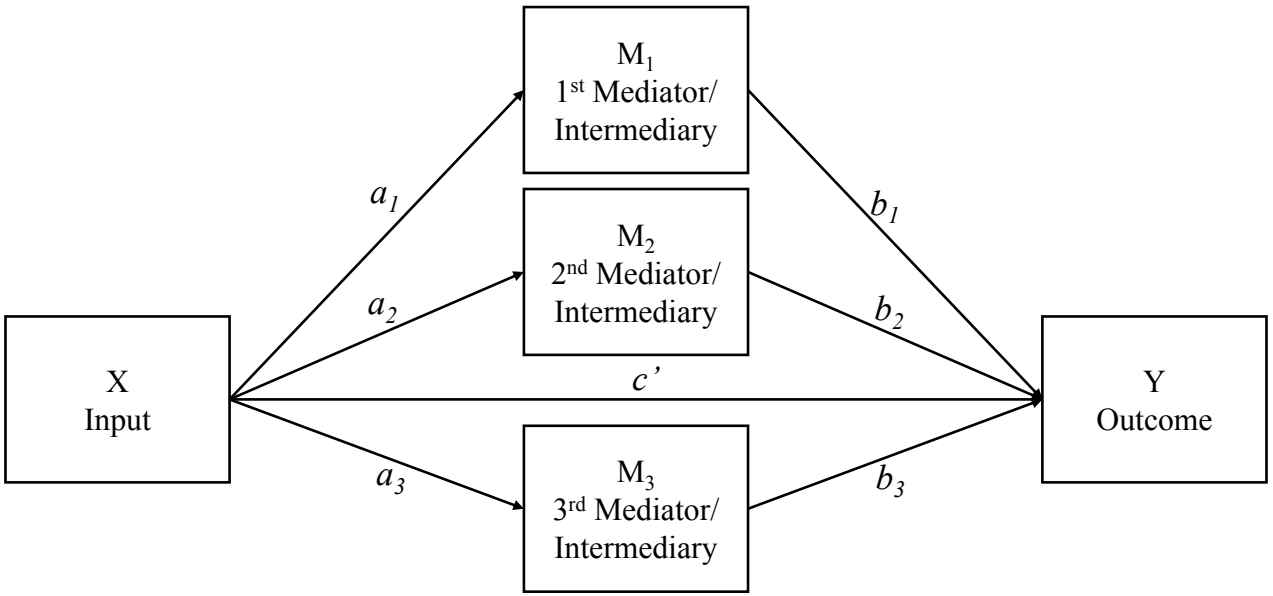
Simplifying our Path Diagram

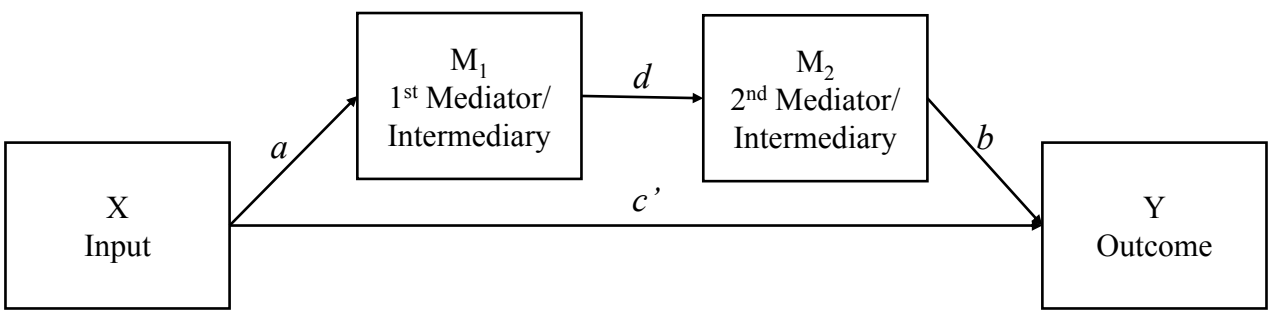
Answer: We don't fit the upper model.



Follow-Up Question: Do we really need to bother?

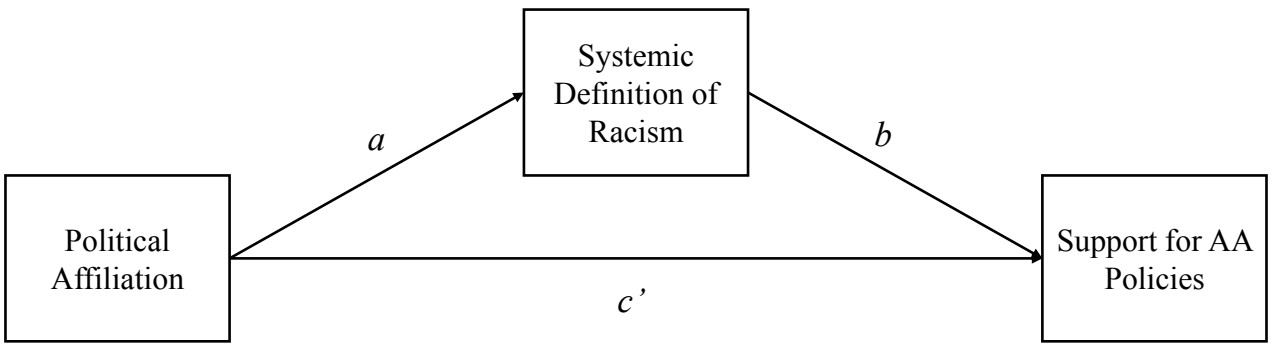
Why Path Analysis?





Example

Let's revisit the example from last week:



We already fit this model with the OLS approach.

Example

```
dat1 ← readRDS("../data/adamsKlpsScaleScore.rds")
## Partial out the mediator's effect:
mod1 ← lm(policy ~ sysRac + polAffil, data = dat1)
mod2 ← lm(sysRac ~ polAffil, data = dat1)
summary(mod1)$coef
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 0.83265885 0.41246491 2.0187386 4.670428e-02
sysRac      0.72235878 0.11147514 6.4799987 5.930291e-09
polAffil    0.05121251 0.06998433 0.7317711 4.663450e-01
```

```
summary(mod2)$coef
```

```
              Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 2.6060451 0.28489391 9.147423 2.715546e-14
polAffil    0.2568494 0.06213495 4.133735 8.336022e-05
```

Example

```
## Extract important parameter estimates:
a ← coef(mod2)["polAffil"]
b ← coef(mod1)["sysRac"]
## Compute indirect effect:
ieProd ← a * b
ieProd
```

```
 polAffil
0.1855374
```

```
## Calculate Sobel's Z:
seA ← sqrt(diag(vcov(mod2)))["polAffil"]
seB ← sqrt(diag(vcov(mod1)))["sysRac"]
sobelSE ← sqrt(b^2 * seA^2 + a^2 * seB^2)
sobelZ ← ieProd / sobelSE
sobelZ
```

```
polAffil
 3.48501
```

Example

```
sobelP ← 2 * pnorm(sobelZ, lower = FALSE)
sobelP
```

```
    polAffil
0.0004921178
```

```
sobelUB ← ieProd + 1.96 * sobelSE
sobelLB ← ieProd - 1.96 * sobelSE
## 95% Sobel CI:
c(sobelLB, sobelUB)
```

```
  polAffil   polAffil
0.08118957 0.28988525
```

Example

```
## Load the lavaan package:
library(lavaan)
## Specify the basic path model:
mod1 <- "
policy ~ sysRac + polAffil
sysRac ~ polAffil
"
## Estimate the model:
out1 <- cfa(mod1, data = dat1)
## Look at the results:
summary(out1)
```

```
lavaan (0.5-20) converged normally after  13 iterations

  Number of observations                            87

  Estimator                                         ML
  Minimum Function Test Statistic                0.000
  Degrees of freedom                                 0

Parameter Estimates:
```

Example

```
  Information                                        Expected
  Standard Errors                                    Standard

Regressions:
                   Estimate   Std.Err   Z-value   P(>|z|)
  policy ~
    sysRac            0.722     0.110     6.595     0.000
    polAffil          0.051     0.069     0.745     0.456
  sysRac ~
    polAffil          0.257     0.061     4.182     0.000

Variances:
                   Estimate   Std.Err   Z-value   P(>|z|)
    policy            0.837     0.127     6.595     0.000
    sysRac            0.802     0.122     6.595     0.000
```

Example

```
## Include the indirect effect:
mod2 <- "
policy ~ b*sysRac + polAffil
sysRac ~ a*polAffil

ab := a*b # Define a parameter for the indirect effect
"
## Estimate the model:
out2 <- cfa(mod2, data = dat1)
## Look at the results:
summary(out2)
```

```
lavaan (0.5-20) converged normally after  13 iterations

  Number of observations                            87

  Estimator                                         ML
  Minimum Function Test Statistic                0.000
  Degrees of freedom                                 0

Parameter Estimates:
```

Example

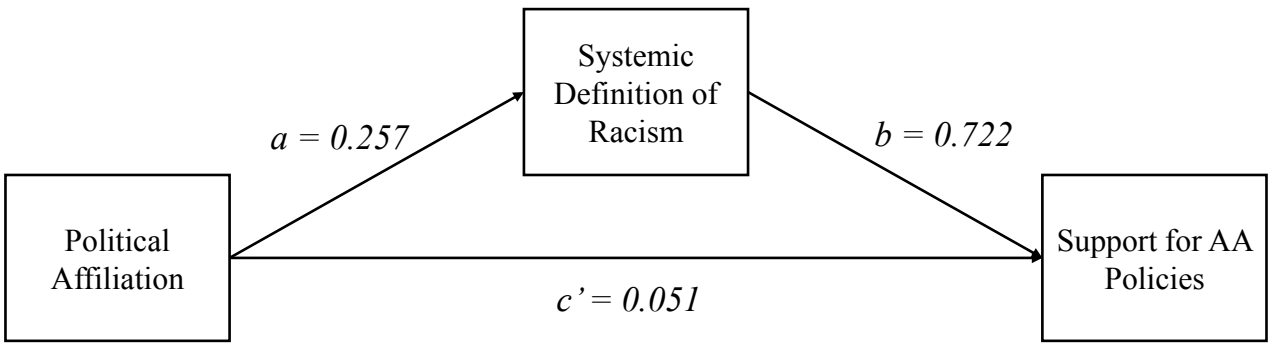
```
  Information                                          Expected
  Standard Errors                                      Standard

Regressions:
                   Estimate   Std.Err   Z-value   P(>|z|)
  policy ~
    sysRac      (b)    0.722     0.110     6.595     0.000
    polAffil           0.051     0.069     0.745     0.456
  sysRac ~
    polAffil    (a)    0.257     0.061     4.182     0.000

Variances:
                   Estimate   Std.Err   Z-value   P(>|z|)
    policy             0.837     0.127     6.595     0.000
    sysRac             0.802     0.122     6.595     0.000

Defined Parameters:
                   Estimate   Std.Err   Z-value   P(>|z|)
    ab                 0.186     0.053     3.532     0.000
```

Example

```
## We can also get CIs:
parameterEstimates(out2)
```

	lhs	op	rhs	label	est	se	z	pvalue	ci.lower	ci.upper
1	policy	~	sysRac	b	0.722	0.110	6.595	0.000	0.508	0.937
2	policy	~	polAffil		0.051	0.069	0.745	0.456	-0.084	0.186
3	sysRac	~	polAffil	a	0.257	0.061	4.182	0.000	0.136	0.377
4	policy	~~	policy		0.837	0.127	6.595	0.000	0.588	1.086
5	sysRac	~~	sysRac		0.802	0.122	6.595	0.000	0.564	1.040
6	polAffil	~~	polAffil		2.444	0.000	NA	NA	2.444	2.444
7	ab	:=	a*b	ab	0.186	0.053	3.532	0.000	0.083	0.289

Results


Systemic Definition of Racism
a = 0.257
b = 0.722
Political Affiliation
Support for AA Policies
c' = 0.051

We're not there yet...

Path analysis allows us to model complex (and simple) patterns of association very parsimoniously, but the preceding example still suffers from a considerable limitation.

We're not there yet...

Path analysis allows us to model complex (and simple) patterns of association very parsimoniously, but the preceding example still suffers from a considerable limitation.

- The significance test for the indirect effect is still conducted with the Sobel Z approach.

Path analysis (or full SEM) doesn't magically get around distributional problems associated with Sobel's Z test.

To get a robust significance test of the indirect effect, we need to use *bootstrapping*.

Bootstrapping

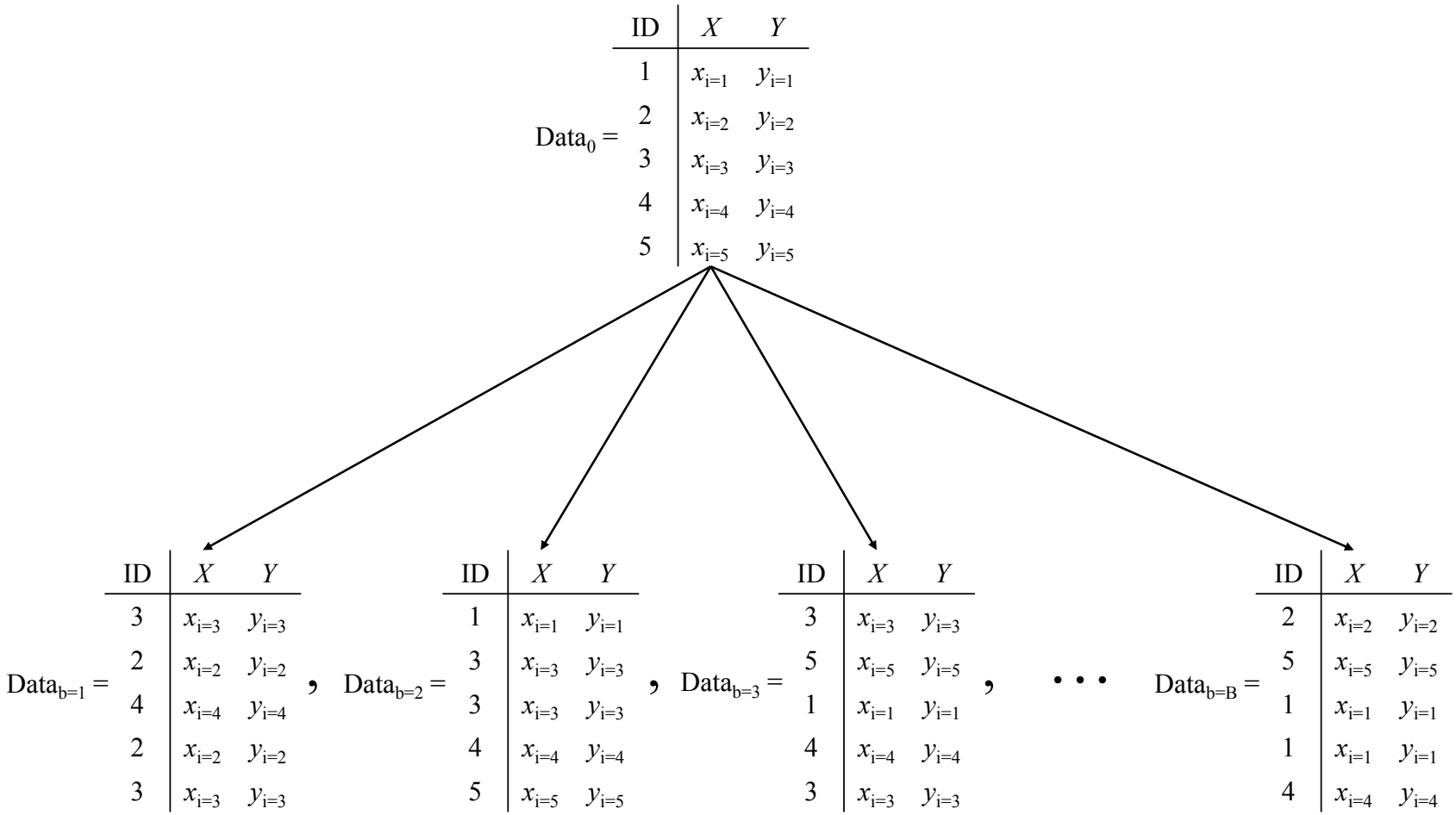
Bootstrapping was introduced by Efron (1979) as a tool for non-parametric inference.

- Traditional inference requires that we assume a parametric sampling distribution for our focal parameter.
- We need to make such an assumption to compute the standard errors we require for inferences.
- If we cannot safely make these assumptions, we can use bootstrapping.

Assume our observed data $Data_0$ represent the population and:

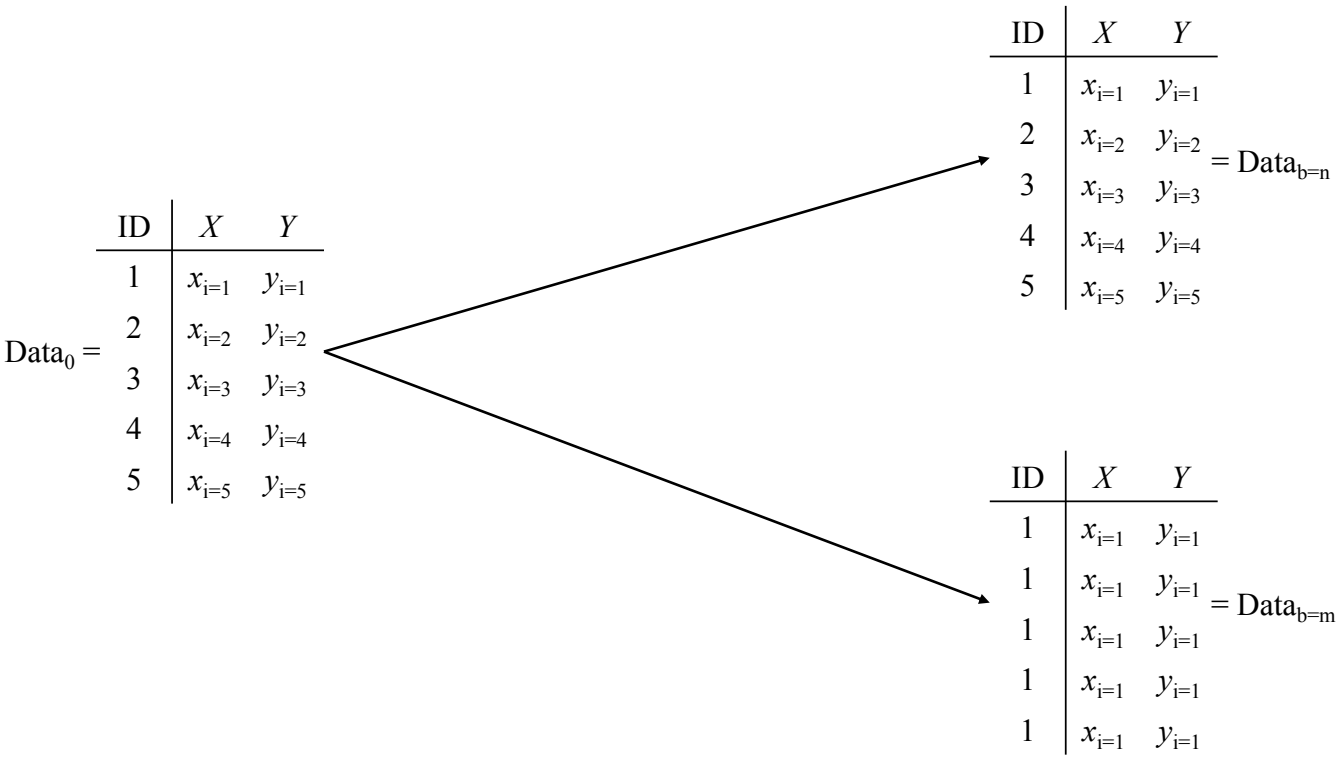
1. Sample rows of $Data_0$, with replacement, to create B new samples $\{Data_b\}$.
2. Calculate our focal statistic on each of the B bootstrap samples.
3. Make inferences based on the empirical distribution of the B estimates calculated in Step 2

Bootstrapping



Bootstrapping

It's important to recognize that the following are legal bootstrap samples:



Example

Suppose I'm on the lookout for a retirement location. Since I want to relax in my old-age, I'm concerned with ensuring a low probability of dragon attacks, so I have a few salient considerations:

- Shooting for a location with no dragons, whatsoever, is a fools errand (since dragons are, obviously, ubiquitous).
- I merely require a location that has at least two times as many dragon-free days as other kinds.

I've been watching several candidate locales over the course of my (long and illustrious) career, and I'm particularly hopeful about one quiet hamlet in the Patagonian highlands.

- To ensure that my required degree of dragon-freeness is met, I'll use the *Dragon Risk Index* (DRI):

$$DRI = Median\left(\frac{\text{Dragon-Free Days}}{\text{Dragonned Days}}\right)$$

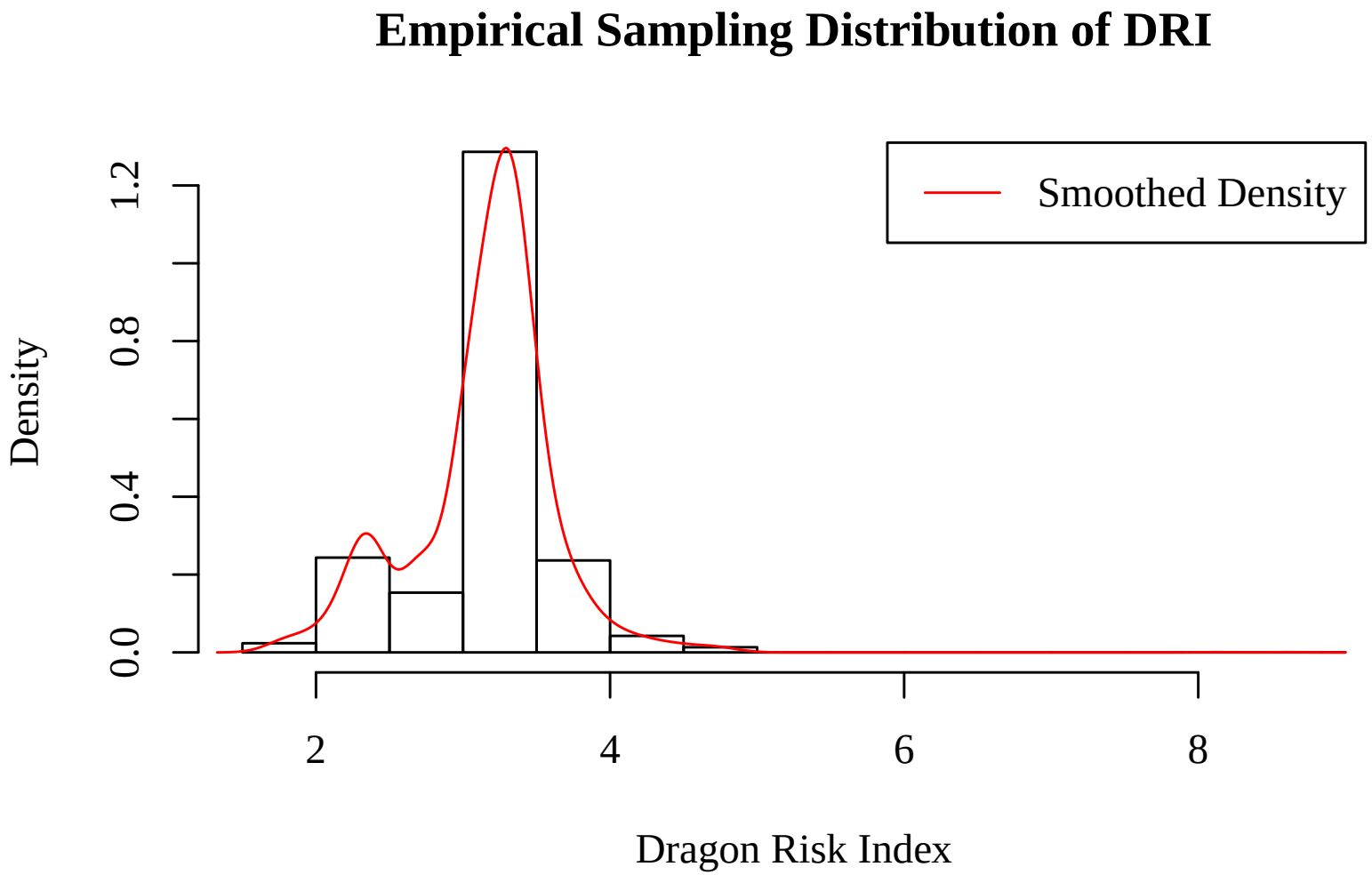
Example

```
## Read in the observed data:
rawData ← readRDS("../data/daysData.rds")
## Compute the observed test statistic:
obsDRI ← median(rawData$goodDays / rawData$badDays)
obsDRI
```

```
[1] 3.24476
```

```
## Draw the bootstrap samples:
set.seed(235711)
nSams ← 5000
bootDRI ← rep(NA, nSams)
for(b in 1 : nSams) {
    bootSam ← rawData[sample(c(1 : nrow(rawData)),
                             replace = TRUE), ]
    bootDRI[b] ←
        median(bootSam$goodDays / bootSam$badDays)
}
```

Example


Empirical Sampling Distribution of DRI
Smoothed Density
Density
0.0
0.4
0.8
1.2
2
4
6
8
Dragon Risk Index

Example

To see if I can be confident in the dragon-freeness of my potential home, I'll summarize the preceding distribution with a (one-tailed) percentile confidence interval:

```
bootLB ← sort(bootDRI)[0.05 * nSams]
bootUB ← Inf
## The bootstrapped Percentile CI:
c(bootLB, bootUB)
```

```
[1] 2.288555      Inf
```

Bootstrapped Path Modeling

We can apply the same procedure I just demonstrated to testing the indirect effect's significance with path modeling.

- The problem with Sobel's Z is exactly the type of issue for which bootstrapping was designed
  - We don't know a reasonable finite-sample sampling distribution for the ab parameter.
- Bootstrapping will allow us to construct an empirical sampling distribution for ab and use that empirical distribution to construct confidence intervals for inference.

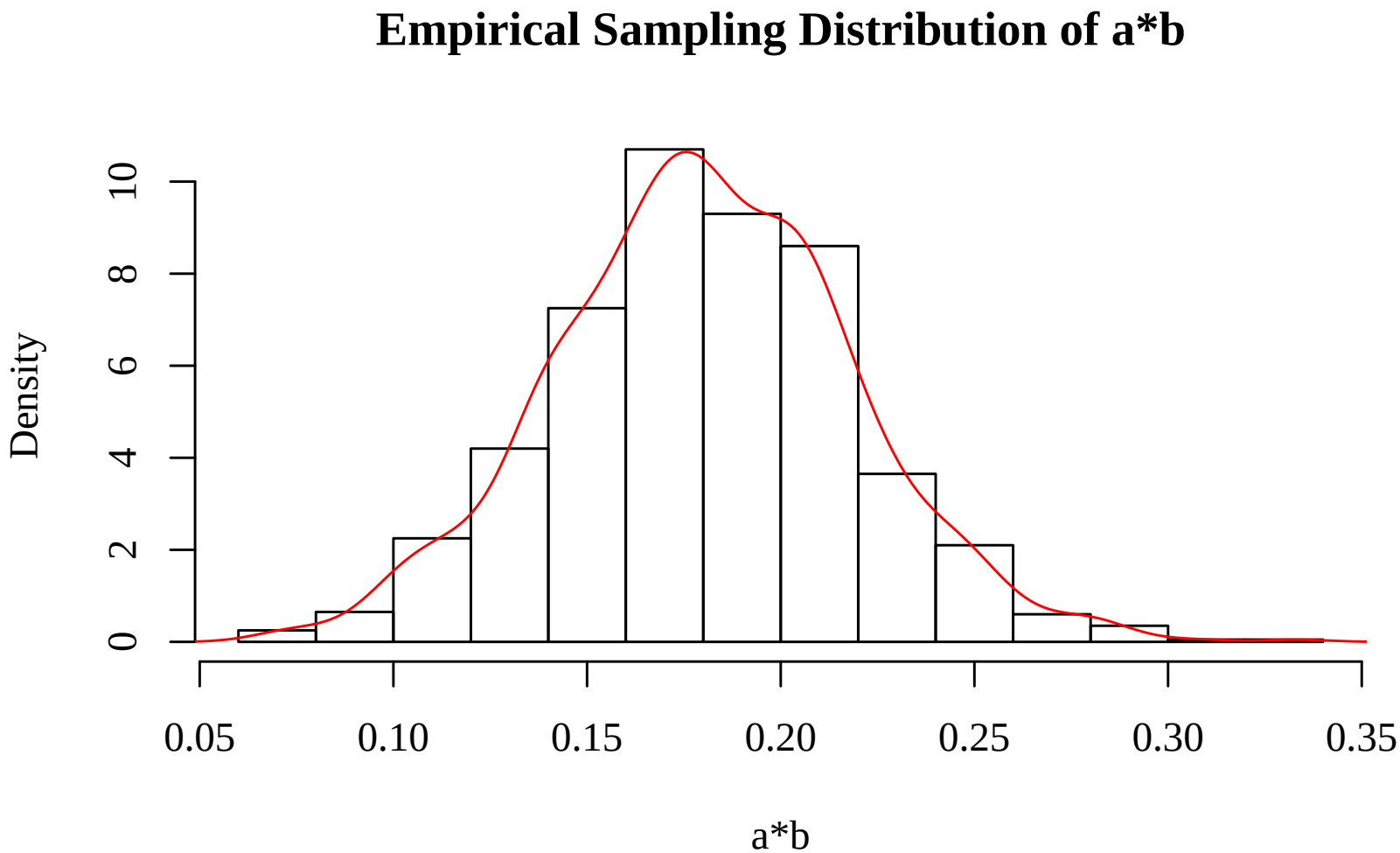
All we need to do is:

1. Resample our observed data with replacement
2. Fit our hypothesized path model to each bootstrap sample
3. Store the value of ab that we get each time
4. Summarize the empirical distribution of ab to make inferences

Example

```
nSams ← 1000
abVec ← rep(NA, nSams)
for(i in 1 : nSams) {
    ## Resample the data:
    bootSam ←
        dat1[sample(c(1 : nrow(dat1)), replace = TRUE), ]
    ## Fit the path model:
    bootOut ← sem(mod2, data = bootSam)
    ## Store the estimated indirect effect:
    abVec[i] ← prod(coef(bootOut)[c("a", "b")])
}
```

Example


Empirical Sampling Distribution of a*b
Density
0
2
4
6
8
10
0.05
0.10
0.15
0.20
0.25
0.30
0.35
a*b

Example

```
## Calculate the percentile CI:
lb ← sort(abVec)[0.025 * nSams]
ub ← sort(abVec)[0.975 * nSams]
c(lb, ub)
```

```
[1] 0.1032923 0.2568537
```

Example

```
## OR, much more parsimoniously:
bootOut2 <- sem(mod2, data = dat1, se = "boot", bootstrap = 1000)
parameterEstimates(bootOut2)
```

	lhs	op	rhs	label	est	se	z	pvalue	ci.lower	ci.upper
1	policy	~	sysRac	b	0.722	0.104	6.921	0.000	0.520	0.923
2	policy	~	polAffil		0.051	0.077	0.664	0.507	-0.111	0.196
3	sysRac	~	polAffil	a	0.257	0.058	4.440	0.000	0.135	0.362
4	policy	~~	policy		0.837	0.177	4.724	0.000	0.517	1.192
5	sysRac	~~	sysRac		0.802	0.127	6.316	0.000	0.543	1.042
6	polAffil	~~	polAffil		2.444	0.000	NA	NA	2.444	2.444
7	ab	:=	a*b	ab	0.186	0.040	4.609	0.000	0.103	0.261

```
parameterEstimates(bootOut2, boot.ci.type = "bca.simple")
```

	lhs	op	rhs	label	est	se	z	pvalue	ci.lower	ci.upper
1	policy	~	sysRac	b	0.722	0.104	6.921	0.000	0.517	0.922
2	policy	~	polAffil		0.051	0.077	0.664	0.507	-0.124	0.183
3	sysRac	~	polAffil	a	0.257	0.058	4.440	0.000	0.137	0.363
4	policy	~~	policy		0.837	0.177	4.724	0.000	0.557	1.295
5	sysRac	~~	sysRac		0.802	0.127	6.316	0.000	0.585	1.095
6	polAffil	~~	polAffil		2.444	0.000	NA	NA	2.444	2.444
7	ab	:=	a*b	ab	0.186	0.040	4.609	0.000	0.114	0.278

Monte Carlo Method/Parametric Bootstrap

We can also use a *parametric bootstrap* of the individual parameters a and b to get a somewhat robust test of the indirect effect.

- Assuming normal sampling distributions for a and b is not, generally, problematic
- We can save ourselves a lot of computational effort by assuming normality for a and b, then:

1. Fit the hypothesized path model to the raw data
2. Extract a, b, and $ACOV(\{a, b\})$ from the fitted path model
3. Parameterize a bivariate normal distribution $\mathrm{N}(a, b|\mu, \Sigma)$ with $\mu = \{a, b\}$ and $\Sigma = ACOV(\{a, b\})$
4. Draw simulated values $\{\tilde{a}, \tilde{b}\}$ from $\mathrm{N}(a, b|\mu, \Sigma)$
5. Compute the simulated indirect effect $\widetilde{ab} = \tilde{a} \cdot \tilde{b}$ and store it
6. Summarize the empirical distribution of $\widetilde{ab}$ for inference.

Example

```
## Load package to draw the Monte Carlo samples
library(mvtnorm)
## Specify the model (note the parameter labels):
mod3 ← "
policy ~ polAffil + b*sysRac
sysRac ~ a*polAffil
"
## Fit the model:
out4 ← sem(mod3, data = dat1)
## Extract the important estimates:
a ← coef(out4)["a"]
b ← coef(out4)["b"]
acov ← vcov(out4)[c("a", "b"), c("a", "b")]
```

Aside Regarding Asymptotic Covariances

The *asymptotic covariance matrix* (ACOV) is (-1 times) the inverse of the Fisher information matrix of the model parameters.

- The *ACOV* contains the expected covariance among the ML estimates of the model parameters.
- The diagonal elements of the matrix (i.e., the asymptotic variances) are the square of the usual ML SE estimates.

```
round(vcov(out4), 4)
```

```
                  plcy~A b       a       plcy~~ syR~~R
policy~polAffil    0.005
b                 -0.003  0.012
a                  0.000  0.000  0.004
policy~~policy     0.000  0.000  0.000  0.016
sysRac~~sysRac     0.000  0.000  0.000  0.000  0.015
```

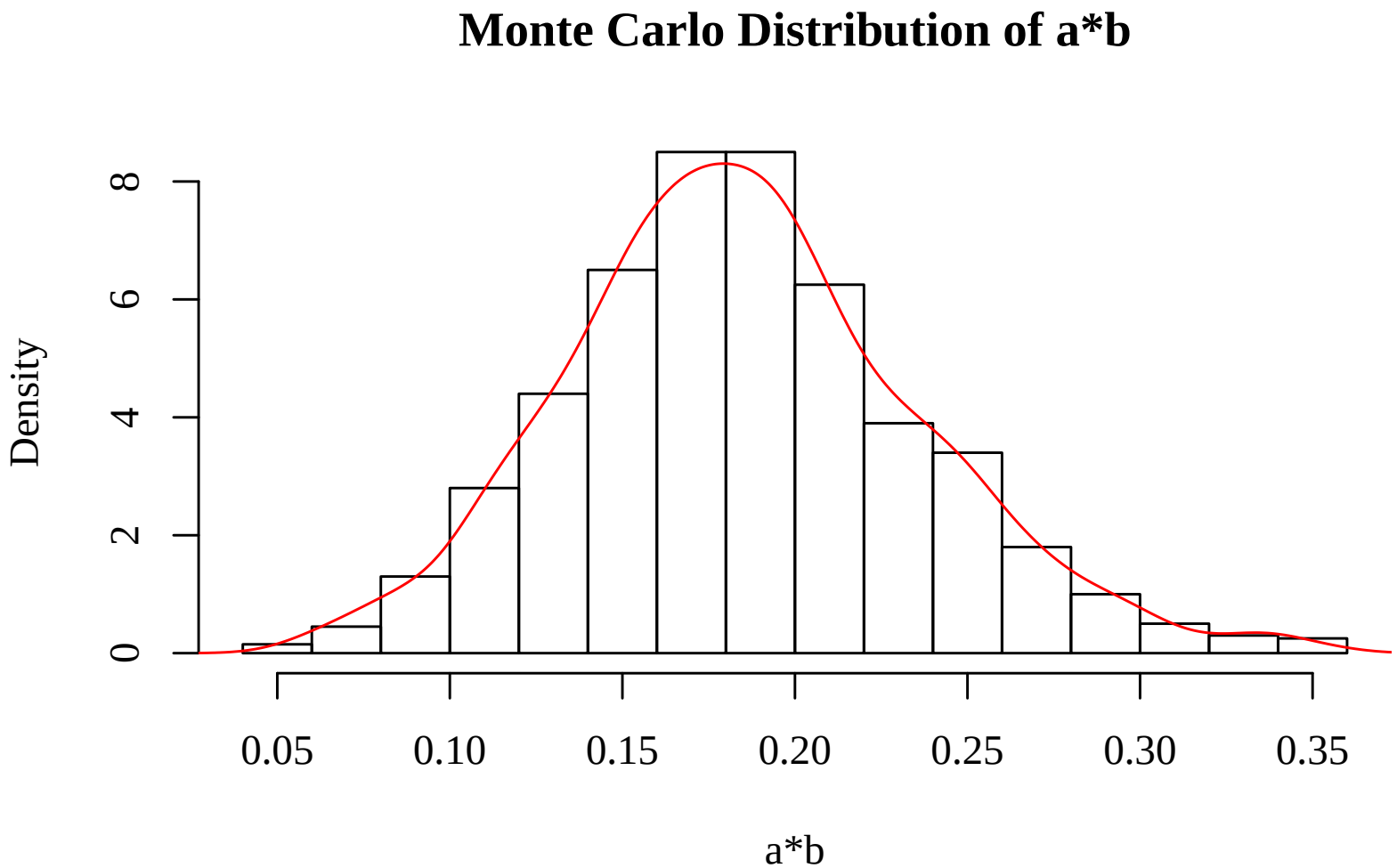
Back to the Example

```
## Look at ACOV(a,b):
round(acov, 4)
```

```
       a      b
a 0.0038 0.000
b 0.0000 0.012
```

```
## Draw the Monte Carlo samples:
samMat ← rmvnorm(nSams, c(a, b), acov)
abVec ← samMat[ , "a"] * samMat[ , "b"]
```

Back to the Example



Back to the Example

```
## Calculate the percentile CI:
lb ← sort(abVec)[0.025 * nSams]
ub ← sort(abVec)[0.975 * nSams]
c(lb, ub)
```

```
[1] 0.08845936 0.29432389
```

Kris Preacher's Website

If you don't want to program the Monte Carlo approach yourself (although each of you easily can), you should consider the very handy Web App on Professor Kris Preacher's website `http://www.quantpsy.org`.

- Kris' website has a vast array of hugely helpful resources for anyone doing mediation or moderation analysis.
- You should definitely check it out!

References

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26. doi: 10.1214/aos/1176344552