An Introduction to Mediation Analysis

Overview

- Introduction to mediation analysis
- Learn how to estimate mediation models
- Learn how to test mediation effects with the bootstrap method
- First glimpse at path analysis

Job Demands-Resources (JDR) Theory: Our Example

The JDR Theory explains how job resources and demands influence one's engagement and burnout which goes onto influence one's job performance:

- Performance feedback affects job performance through its relationship with engagement
- Social support affects job performance through its relationship with burnout and engagement

What is Mediation Analysis?

Mediation analysis is a statistical method used to test a set of relationships, causal or predictive, where one variable, **the antecedent variable**, influences an outcome variable **indirectly** through its influence on a mediating variable referred to as a **mediator**.

Some Mediation Analysis Jargon

Like all things statistical, mediation analysis comes with its own set of jargon:

- Antecedent Variable
- Mediator or Mediating Variable
- Mechanism
- Path Analysis & Path Model
- Total, Direct, & Indirect Effects
- Simple vs Multiple Mediation

Is Mediation Analysis Important to Know?

Yes.

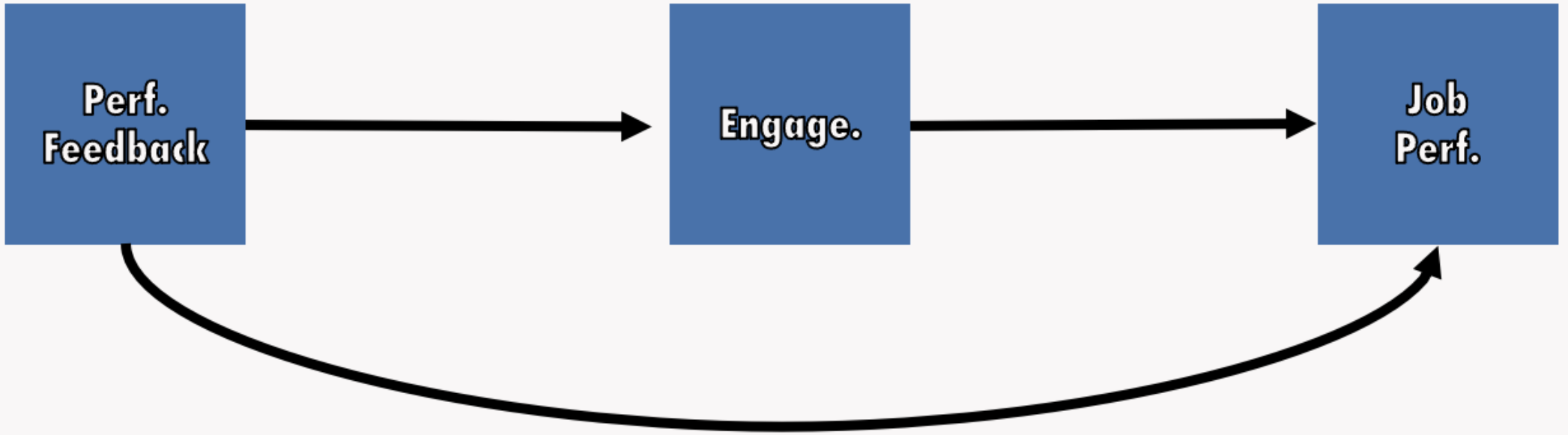
Mediation is present in everywhere and all interesting theories propose some model where mediation is present.

Mediation Effects: Total, Indirect, and Direct Effects

Mediation analysis allows us to decompose the total effect that the antecedent variable has on the outcome variable into two pieces:

- **Direct Effect**: The effect the antecedent variable has on the outcome variable, while controlling for the mediator.
- **Indirect Effect**: The effect the antecedent has on the outcome variable through the mediator.
- **Total Effect**: The sum of the direct and indirect effects. It is the effect of the antecedent variable on the outcome variable without controlling for the mediator.

Path Model: Performance Feedback to Engagement to Job Performance



The Algebra of Mediation: Estimating the Total Effect

You can estimate the total effect by regressing the outcome variable, Y, onto the antecedent variable, X.

$$Y = \beta_0 + \beta_1 X$$

The Algebra of Mediation: Estimating the Direct Effect

You can estimate the direct effect of the antecedent variable on the outcome variable by regressing the outcome variable, Y, onto both the antecedent variable and mediator.

$$Y = \beta_0 + \beta_2 X + \beta_3 M$$

The Algebra of Mediation: Estimating Components of the Indirect Effects

By regressing the outcome variable onto the mediator and the mediator onto the antecedent variable, you estimate the two component pieces that make up the indirect effect: β_3 & β_4.

$$Y = \beta_0 + \beta_2 X + \beta_3 M$$

$$M = \beta_0 + \beta_4 X$$

The Algebra of Mediation: Calculating the Indirect & Total Effects

You can calculate the indirect effect by multiplying together its component pieces, the effect of X on M and the effect of M on Y. You can calculate the total effect by adding the direct and indirect effects.

$$Y = \beta_0 + \beta_2 X + \beta_3 M$$

$$M = \beta_0 + \beta_4 X$$

$$\text{Indirect Effect} = \beta_4 \times \beta_3$$

$$\text{Total Effect} = \beta_2 + \beta_4 \times \beta_3$$

Estimating a Mediation Model with Linear Regression: Performance Feedback

We can use the following regression models to test for simple mediation:

```
mod_total <- lm(job_perf ~ perf_feedback, data = data_mediator)
mod_de <- lm(job_perf ~ engagement + perf_feedback, data = data_mediator)
mod_ie <- lm(engagement ~ perf_feedback, data = data_mediator)
```

Interpretation & Inference of the Total Effect

We can interpret and draw inferences about the total effect just like we do any regression effect:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.004	0.037	-0.116	0.908
perf_feedback	0.197	0.035	5.585	0.000

Interpretation & Inference of the Direct Effect

We can interpret and draw inferences about the direct effect just like we do any regression effect:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.017	0.036	-0.473	0.637
engagement	0.258	0.048	5.352	0.000
perf_feedback	0.105	0.039	2.703	0.007

Calculating the Indirect Effect

To calculate the indirect effect, we need to multiply the regression coefficient that captures the effect of performance feedback on engagement by the regression coefficient that captures the effect of engagement on job performance:

```
mod_de$coefficients["engagement"] * mod_ie$coefficients["perf_feedback"]
```

```
engagement
     0.092
```

Interpreting the Indirect Effect

For every one unit increase in performance feedback, job performance changes by 0.092 as a result of performance feedback's effect on engagement which, in turn, affects job performance.

How Do We Make Inferences About the Indirect Effect?

Making inferences about the indirect effect is more challenging as the indirect effect is the **product** of two regression coefficients. This presents two challenges:

1. We can only approximate the standard error
2. The distribution of a product term (like the indirect effect) is **not normal**

Methods to Make Inferences About the Indirect Effect

We have a few methods available to make inferences about indirect effects:

- Normal Theory (Distribution) Approach
- Bootstrap Confidence Intervals
- Monte Carlo Confidence Intervals

Inference About the Indirect Effect: Normal Theory Approach

The Normal Theory approach:

1. Calculates an approximate standard error for the indirect effect
2. Calculates a test statistic by dividing the indirect effect by the approximate standard error
3. Uses the normal distribution to determine the p-value associated with the test statistic

Normal Theory: Approximate Standard Error

$$\mathrm{SE}(\beta_4\beta_3) = \sqrt{\beta_4^2\mathrm{SE}^2(\beta_3) + \beta_3^2\mathrm{SE}^2(\beta_4)}$$

Normal Theory: Example

```r
b3 <- mod_de$coefficients["engagement"]
b4 <- mod_ie$coefficients["perf_feedback"]

se_b3 <- summary(mod_de)$coefficients["engagement", "Std. Error"]
se_b4 <- summary(mod_ie)$coefficients["perf_feedback", "Std. Error"]

se_ie <- sqrt(b4^2 * se_b3^2 + b3^2 * se_b4^2)

test_stat <- (b3 * b4) / se_ie

p_value <- pnorm(test_stat, lower.tail = FALSE) * 2
```

Indirect Effect	SE	Test Stat.	p value
0.092	0.018	5.063	0

Inference About the Indirect Effect: Bootstrap Confidence Intervals

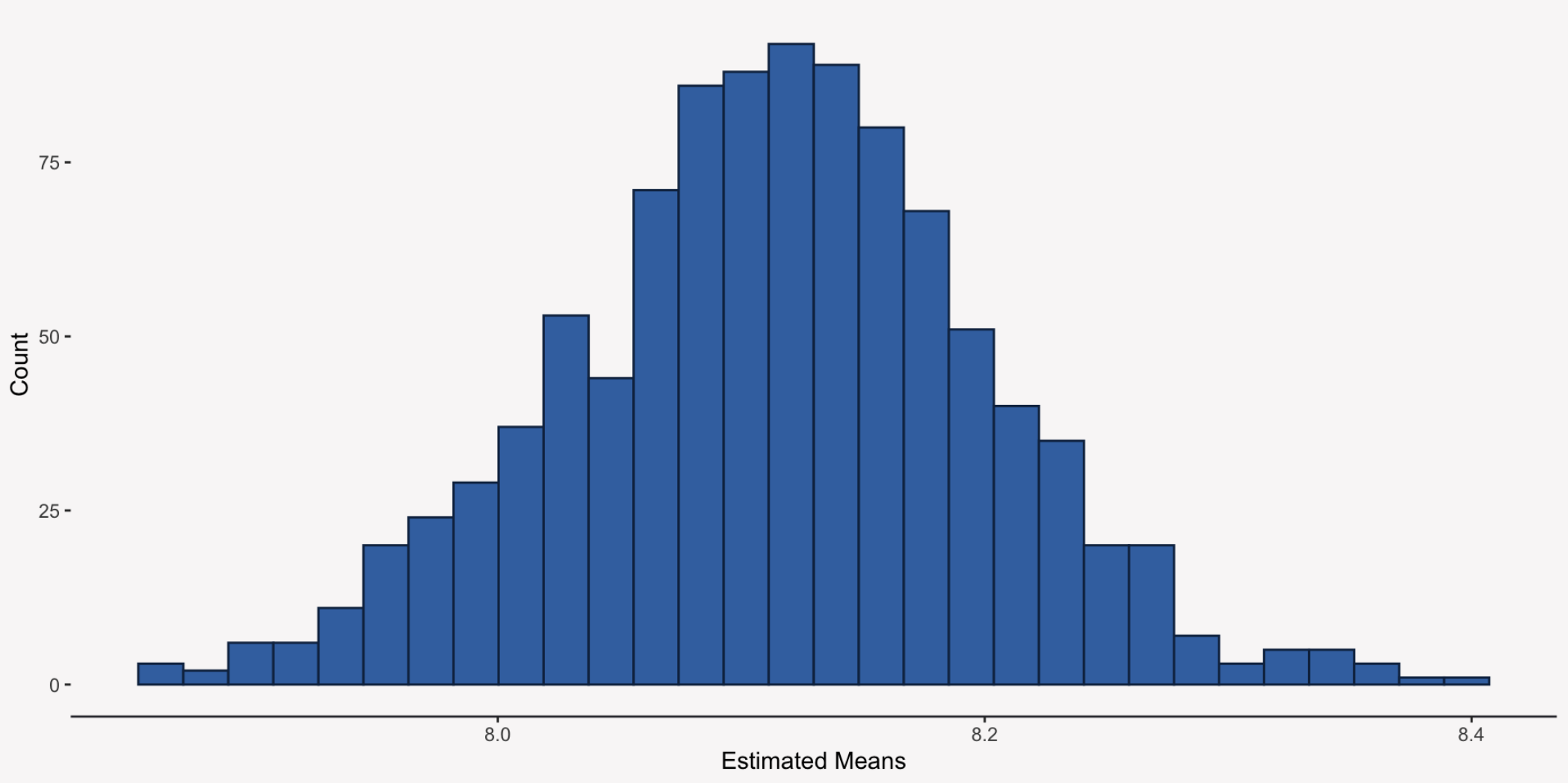
The bootstrap method is a computationally intensive way to calculate confidence intervals for any statistic (e.g. regression slope or indirect effect) by:

1. Randomly sample with replacement *N* rows from your dataset.
2. Estimate your statistic from your sampled data and save it.
3. Repeat this process **A LOT** of times (at least 1,000 times).
4. Use the distribution of your saved estimates to build a confidence interval.

Coding a Bootstrap Intervals for the Mean

```r
set.seed(1)
x <- rnorm(100, mean = 8, sd = 1)
saved_mean <- numeric(1000)

for(i in 1:1000) {

  sample_rows <- sample(1:length(x), size = length(x), replace = TRUE)
  new_data <- x[sample_rows]
  saved_mean[i] <- mean(new_data)

}
```

Coding Bootstrap Confidence Intervals for the Mean



Coding Bootstrap Confidence Intervals for the Indirect Effect

Instead of writing our own bootstrap function, we can use the `boot` function from the `boot` package.

First we have to write a function that estimates our statistic:

```
calc_ie <- function(data, indices, formula_de, formula_m,
                    x_name, m_name) {

  d <- data[indices, ]
  mod_de <- lm(formula_de, data = d)
  mod_m <- lm(formula_m, data = d)

  ie <- mod_de$coef[m_name] * mod_m$coef[x_name]

  return(ie)

}
```

Coding Bootstrap Intervals for the Indirect Effect

Then we provide our function to the `boot` function along with some additional arguments such as the number of bootstraps to take, `R = 1000`:

```
set.seed(54)

ie_boot <- boot::boot(
  data = data_mediator,
  statistic = calc_ie,
  R = 1000,
  formula_m = engagement ~ perf_feedback,
  formula_de = job_perf ~ engagement + perf_feedback,
  x_name = "perf_feedback",
  m_name = "engagement"
)
```

Coding Bootstrap Intervals for the Indirect Effect

Finally, we can use the `boot.ci` function to calculate the bootstrap confidence intervals:

```
1  boot::boot.ci(ie_boot, conf = .95, type = "perc")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot::boot.ci(boot.out = ie_boot, conf = 0.95, type = "perc")

Intervals :
Level     Percentile
95%   ( 0.0585,  0.1282 )
Calculations and Intervals on Original Scale
```

Inference About the Indirect Effect: Monte Carlo Confidence Intervals

We can use the estimated regression coefficients and their standard errors to simulate a normal sampling distribution:

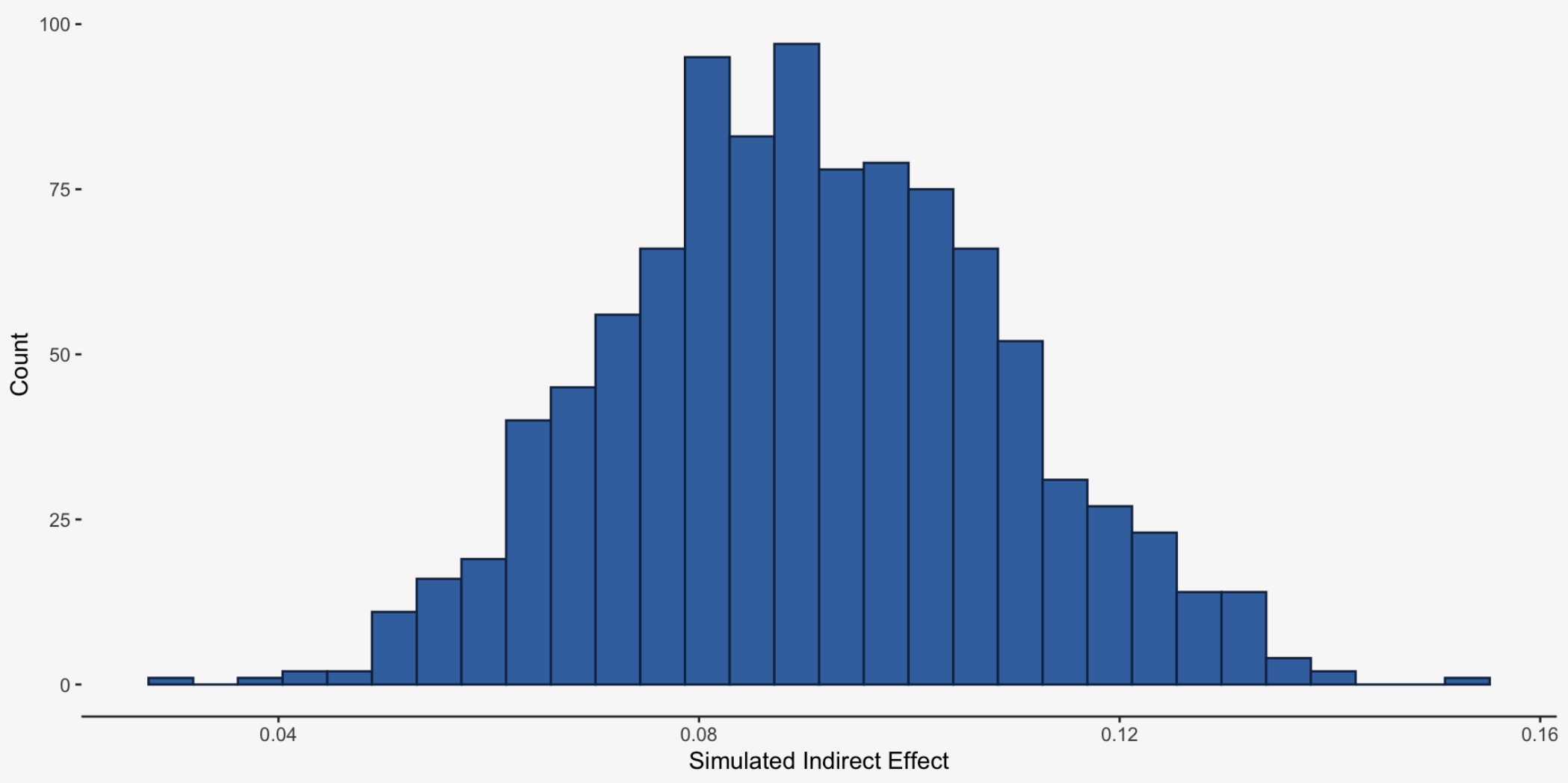
1. Estimate the necessary linear regression models.
2. Save the necessary estimates from the linear models.
3. Use an algorithm such as `rnorm` to generate a lot of observations with a mean equal to the estimated regression coefficient and standard deviation equal to the estimated standard error.
4. Do step 3 for both β_3 and β_4 and multiply each observation together to create a sampling distribution for the indirect effect: $\beta_3 \beta_4$.

Coding Monte Carlo Intervals

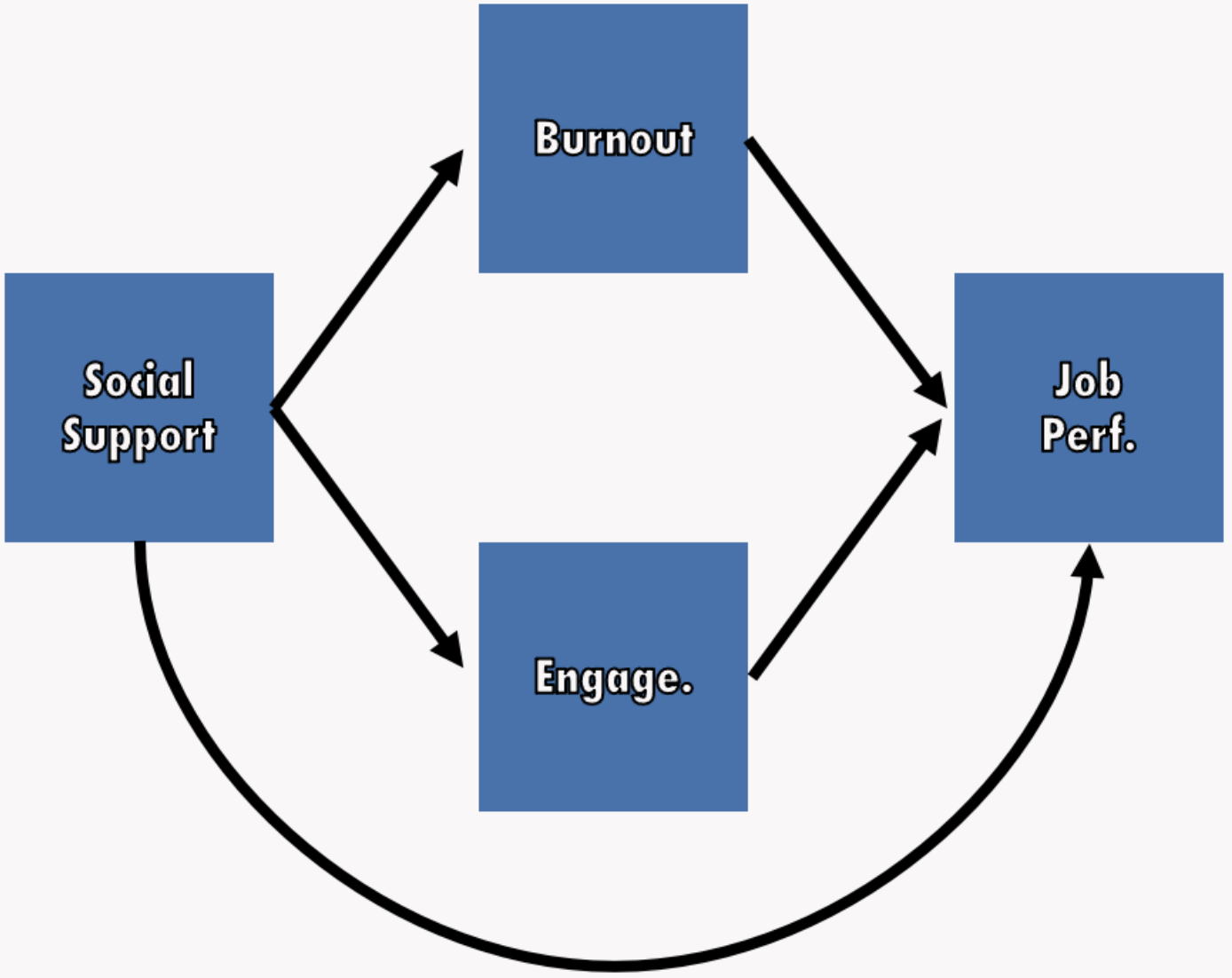
```r
norm_b3 <- rnorm(1000, mean = b3, sd = se_b3)
norm_b4 <- rnorm(1000, mean = b4, sd = se_b4)
norm_ie <- norm_b3 * norm_b4

quantile(norm_ie, c(.025, .975))
```

```
      2.5%       97.5% 
0.05530038 0.12792893 
```

Coding Monte Carlo Intervals



Multiple Mediator Model: Influence of Social Support through Burnout and Engagement



Estimating a Multiple Mediator Model with Linear Regression

We can use the following regression models to test for simple mediation:

```r
mod_total <- lm(job_perf ~ social_supp, data = data_mediator)
mod_de <- lm(job_perf ~ engagement + burnout + social_supp, data = data_mediator)
mod_ie_engage <- lm(engagement ~ social_supp, data = data_mediator)
mod_ie_burnout <- lm(burnout ~ social_supp, data = data_mediator)
```

Interpretation & Inference of the Total Effect

We can interpret and draw inferences about the total effect just like we do any regression effect:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.002	0.036	0.046	0.963
social_supp	0.237	0.037	6.481	0.000

Interpretation & Inference of the Direct Effect

We can interpret and draw inferences about the direct effect just like we do any regression effect:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.019	0.034	-0.574	0.566
engagement	0.286	0.041	6.985	0.000
burnout	-0.358	0.032	-11.028	0.000
social_supp	0.030	0.038	0.789	0.431

Calculating the Indirect Effects

Now we have two indirect effects to calculate:

1. Indirect effect of social support through engagement
2. Indirect effect of social support through burnout

```
mod_de$coefficients["engagement"] * mod_ie_engage$coefficients["social_supp"]
mod_de$coefficients["burnout"] * mod_ie_burnout$coefficients["social_supp"]
```

```
engagement
     0.041
```

```
burnout
  0.167
```

Inference about the Indirect Effects: Bootstrap Confidence Intervals

To calculate the 95% confidence intervals for each indirect effect, we will use the bootstrap method as it is typically the most accurate compared to the other two methods.

Coding the Bootstrap Confidence Interval

```r
calc_ie <- function(data, indices, formula_de, formula_m1, formula_m2,
                    x_name, m1_name, m2_name) {

  d <- data[indices, ]
  mod_de <- lm(formula_de, data = d)
  mod_m1 <- lm(formula_m1, data = d)
  mod_m2 <- lm(formula_m2, data = d)

  ie_m1 <- mod_de$coef[m1_name] * mod_m1$coef[x_name]
  ie_m2 <- mod_de$coef[m2_name] * mod_m2$coef[x_name]

  results <- c(ie_m1, ie_m2)

  return(results)

}
```

Coding Bootstrap Intervals for the Indirect Effect

Then we provide our function to the `boot` function along with some additional arguments such as the number of bootstraps to take, `R = 1000`:

```
set.seed(674)

ie_boot <- boot::boot(
  data = data_mediator,
  statistic = calc_ie,
  R = 1000,
  formula_m1 = engagement ~ social_supp,
  formula_m2 = burnout ~ social_supp,
  formula_de = job_perf ~ engagement + burnout + social_supp,
  x_name = "social_supp",
  m1_name = "engagement",
  m2_name = "burnout"
)
```

Coding Bootstrap Intervals for the Indirect Effect

Finally, we can use the `boot.ci` function to calculate the bootstrap confidence intervals for each indirect effect:

```
boot::boot.ci(ie_boot, conf = .95, type = "perc", index = 1)
boot::boot.ci(ie_boot, conf = .95, type = "perc", index = 2)
```

Confidence Intervals for the Indirect Effect Through Engagement

```r
boot::boot.ci(ie_boot, conf = .95, type = "perc", index = 1)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot::boot.ci(boot.out = ie_boot, conf = 0.95, type = "perc",
    index = 1)

Intervals :
Level     Percentile
95%   ( 0.0251,  0.0605 )
Calculations and Intervals on Original Scale
```

Confidence Intervals for the Indirect Effect Through Burnout

```r
boot::boot.ci(ie_boot, conf = .95, type = "perc", index = 2)
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot::boot.ci(boot.out = ie_boot, conf = 0.95, type = "perc",
    index = 2)

Intervals :
Level     Percentile
95%   ( 0.1304,  0.2040 )
Calculations and Intervals on Original Scale
```

Causality & Mediation

Mediation is inherently causal. It is very tricky to talk about mediation without invoking causality. Because most of you likely will not be able to conduct a true experiment, here are a couple pointers on arguing for causality:

1. You must be able to make a strong argument that X occurs before M, in time, and M occurs before Y.
2. You will need to have a strong theory underlying your mediation model.

Advances in Mediation

There are many new advances in mediation analysis. Here are a few:

- Categorical Antecedent Variables
- Categorical Mediator Variables
- Moderated Mediation

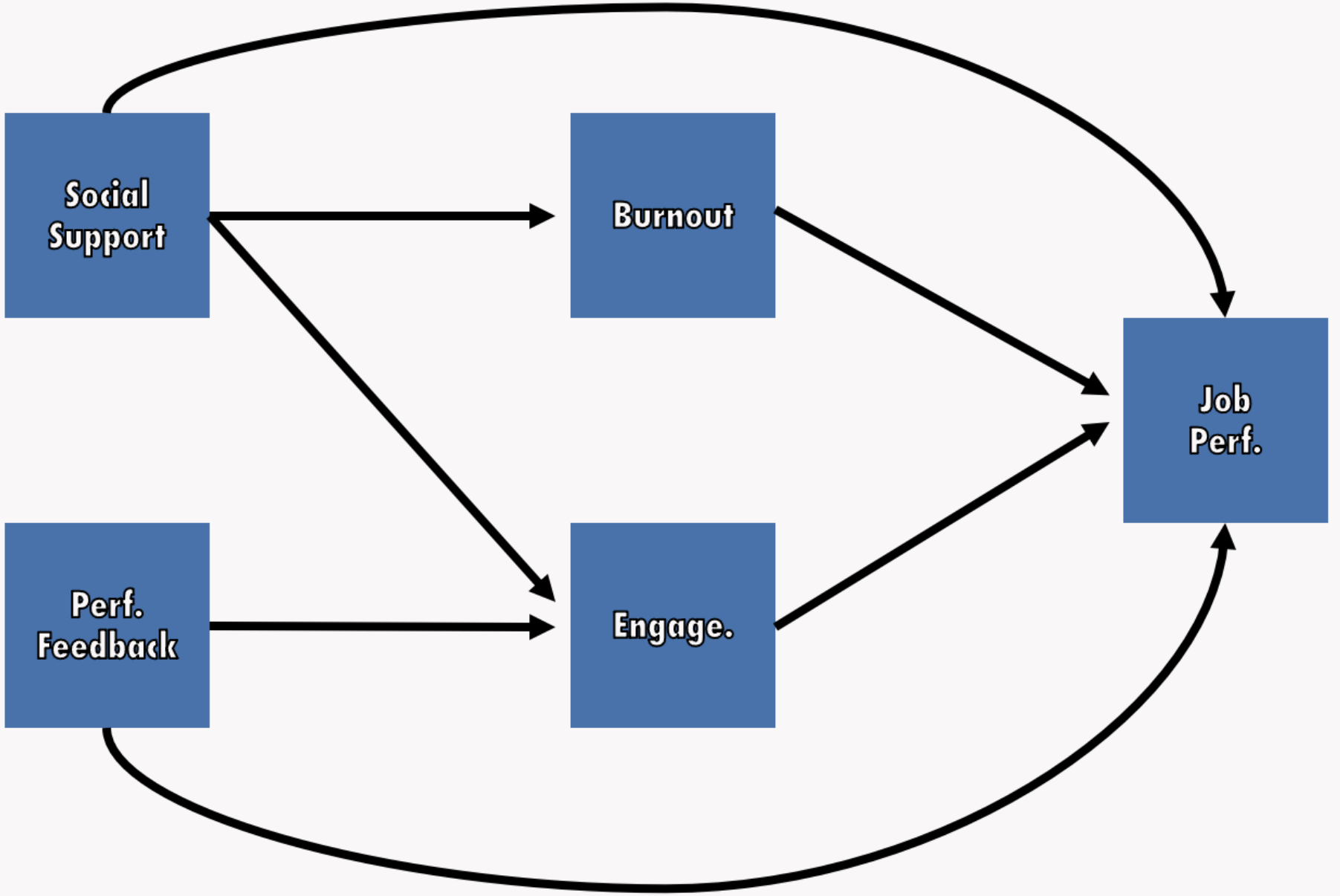
Path Analysis with Structural Equation Models (SEM)

What is Path Analysis with SEM?

Path analysis is really no different from linear regression. But instead of a single regression equation, path analysis works with a set (system) of regression equations.

SEM is a linear method that allows us to fit our path model all at once rather than estimate each regression equation separately.

Full Path Model



Building a Path Model with `lavaan`

```
path_model <- '
engagement ~ m_ss_en*social_supp + m_pf_en*perf_feedback
burnout ~ m_ss_bo*social_supp
job_perf ~ de_en*engagement + de_bo*burnout + social_supp + perf_feedback

ie_ss_en := m_ss_en * de_en
ie_ss_bo := m_ss_bo * de_bo
ie_pdf_en := m_pf_en * de_en
'
```

Estimating our Path Model

```
path_model_est <- lavaan::sem(path_model, data = data_mediator)
```

Interpreting a Path Model

```
1 summary(path_model_est)
```

```
lavaan 0.6.16 ended normally after 1 iteration

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        10

  Number of observations                          1000

Model Test User Model:

  Test statistic                                 0.567
  Degrees of freedom                                 2
  P-value (Chi-square)                           0.753

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
```