mediation_graphs

```r
# Load required packages
library(MASS)
library(lavaan)
```

```
This is lavaan 0.6-19
lavaan is FREE software! Please report any bugs.
```

```r
library(glue)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following object is masked from 'package:MASS':

    select

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(parallel)
```

```r
rsquare_med <- function(data, x, m, y) {
  # Compute correlations among the variables
  rxm <- cor(data[x], data[[m]])
  rxy <- cor(data[[x]], data[[y]])
  rmy <- cor(data[[m]], data[[y]])

  # Regression: m ~ x (to get alpha, first indirect path)
  # Equation 2 in Fairchild, et al
  model1 <- lm(as.formula(paste(m, "~", x)), data = data)
  alpha <- coef(model1)[[x]]

  # Regression: y ~ x + m (to get 'tau_prime' and 'beta')
  # Equation 1 in Fairchild, et al
  model2 <- lm(as.formula(paste(y, "~", x, "+", m)), data = data)

  tau_prime <- coef(model2)[[x]]
  beta <- coef(model2)[[m]]

  # Compute total effect of x on y: tau = tau_prime + (alpha*beta)
  total <- tau_prime + (alpha*beta)


  # Compute effect-size measures
  mediatedeffect <- alpha * beta       # Indirect effect of x on y via M = alpha*beta
  rxmsquared <- rxm^2                  # squared correlation between x and m
  partialrxy_msquared <- ((rxy - rmy * rxm) / sqrt((1 - rmy^2) * (1 - rxmsquared)))^2
  partialrmy_xsquared <- ((rmy - rxy * rxm) / sqrt((1 - rxy^2) * (1 - rxmsquared)))^2
  overallrsquared <- (((rxy^2) + (rmy^2)) - (2 * rxy * rmy * rxm)) / (1 - rxmsquared)
  rsquaredmediated <- (rmy^2) - (overallrsquared - (rxy^2))
  proportionmediated <- if (total != 0) mediatedeffect / total else NA

  # Create a list of results
  results <- list(
    alpha = alpha,
    beta = beta,
    tau_prime = tau_prime,
    total = total,
    mediatedeffect = mediatedeffect,
    rxm = rxm,
    rxmsquared = rxmsquared,
    rxy = rxy,
```

```
    rmy = rmy,
    partialrxy_msquared = partialrxy_msquared,
    partialrmy_xsquared = partialrmy_xsquared,
    overallrsquared = overallrsquared,
    rsquaredmediated = rsquaredmediated,
    proportionmediated = proportionmediated
  )

  return(results)
}

# Define a simulation function that takes an indirect effect value and a condition.
simulate_indirect_condition <- function(indirect, condition, sample_size = 1000, num_reps

  # Set parameters and assign a condition label based on the input.
  if (condition == "blue") {
    # Blue condition: alpha = 1, beta = indirect --> Label: "a = 1"
    pop_alpha  <- 1
    pop_beta   <- indirect
    cond_label <- "a = 1"
  } else if (condition == "red") {
    # Red condition: alpha = indirect, beta = 1 --> Label: "b = 1"
    pop_alpha  <- indirect
    pop_beta   <- 1
    cond_label <- "b = 1"
  } else {
    stop("Unknown condition. Choose 'blue' or 'red'.")
  }

  pop_tau_prime <- 0  # No direct effect

  # Create a "fake" dataset (used only to define the lavaan model)
  d_fake <- data.frame(
    x = rnorm(sample_size),
    m = rnorm(sample_size),
    y = rnorm(sample_size)
  )

  # Build the lavaan model string
  model_string <- glue("
    # Equation for y: note the direct effect of x is set to 0
    y ~ {pop_tau_prime}*x + {pop_beta}*m
```

```
  # Equation for m
  m ~ {pop_alpha}*x
  # Fix variances to 1
  x ~~ 1*x
  y ~~ 1*y
  m ~~ 1*m
")

# Fit the model using lavaan to extract the implied covariance matrix
fit <- lavaan::lavaan(model = model_string, data = d_fake)
pop_cov <- lavaan::lavInspect(fit, "cov.all")

# Generate a "population" dataset (empirical = TRUE)
pop_data <- as.data.frame(
  MASS::mvrnorm(n = sample_size,
                mu = rep(0, 3),
                Sigma = pop_cov,
                empirical = TRUE)
)

# Compute the "true" values using your rsquare_med() function
pop_rs <- rsquare_med(data = pop_data, x = "x", m = "m", y = "y")

# Run simulation replications
sim_matrix <- replicate(num_reps, {
  sim_data <- as.data.frame(
    MASS::mvrnorm(n = sample_size,
                  mu = rep(0, 3),
                  Sigma = pop_cov,
                  empirical = FALSE)
  )
  unlist(rsquare_med(data = sim_data, x = "x", m = "m", y = "y"))
})

# Calculate the average estimates over replications
sim_means <- rowMeans(sim_matrix)
sim_means["proportionmediated"] <- if (sim_means["total"] != 0) {
  sim_means["mediatedeffect"] / sim_means["total"]
} else NA

# Return a data frame with the results and the condition label
```

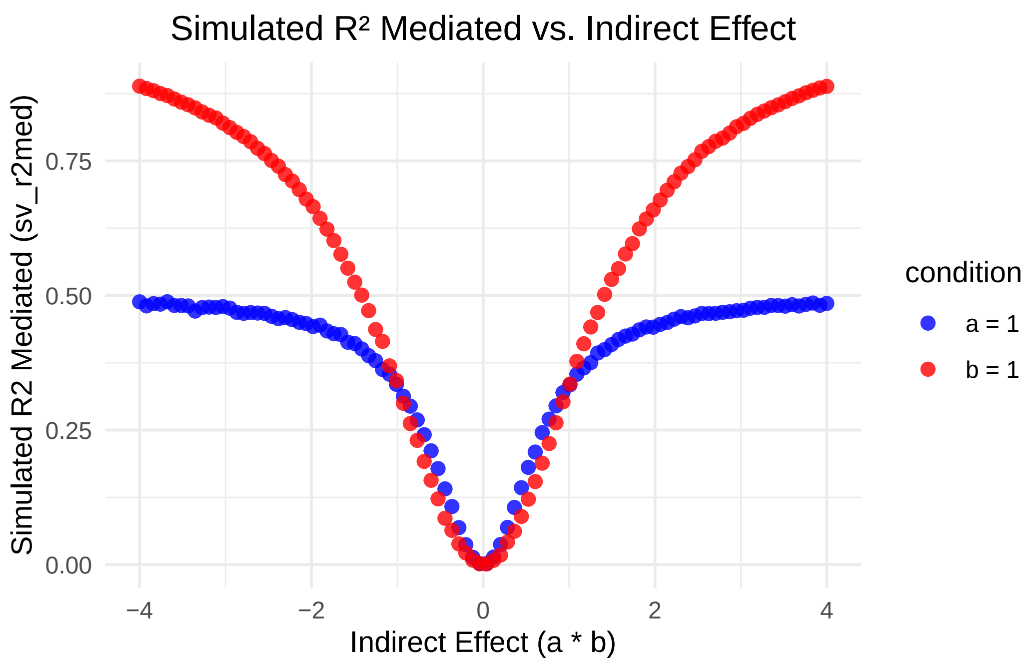
```r
  data.frame(
    indirect_effect = indirect,
    sv_r2med        = sim_means["rsquaredmediated"],
    pop_r2med       = pop_rs$rsquaredmediated,
    condition       = cond_label,
    stringsAsFactors = FALSE
  )
}

# Create a grid of indirect effect values from -4 to 4.
indirect_values <- seq(-4, 4, length.out = 100)

# Run simulations for each condition:
# "blue" will produce condition label "a = 1"
a_results <- lapply(indirect_values, function(x) {
  simulate_indirect_condition(indirect = x, condition = "blue",
                              sample_size = 1000, num_reps = 100)
})
# "red" will produce condition label "b = 1"
b_results <- lapply(indirect_values, function(x) {
  simulate_indirect_condition(indirect = x, condition = "red",
                              sample_size = 1000, num_reps = 100)
})

# Combine the simulation results
sim_results_indirect <- bind_rows(a_results, b_results)

# Create the scatterplot with custom colors and condition labels in the legend.
ggplot(sim_results_indirect, aes(x = indirect_effect, y = sv_r2med, color = condition)) +
  geom_point(size = 2, alpha = 0.8) +
  scale_color_manual(values = c("a = 1" = "blue", "b = 1" = "red")) +
  labs(x = "Indirect Effect (a * b)",
       y = "Simulated R2 Mediated (sv_r2med)",
       title = "Simulated R² Mediated vs. Indirect Effect") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
simulate_random_model <- function(sample_size = 1000, num_reps = 100) {
  # Randomly draw   and   from a normal distribution
  pop_alpha <- rnorm(1)
  pop_beta  <- rnorm(1)
  total_effect <- 0.2
  #pop_alpha <- runif(1, min=-2, max=2)
  #pop_beta <- runif(1, min=-2, max=2)

  # Compute the indirect effect (  *  )
  indirect_effect <- pop_alpha * pop_beta

  # Adjust   (tau prime) so that the total effect (tau_prime +   *  ) equals total_effect
  pop_tau_prime <- total_effect - indirect_effect

  # Create a "fake" dataset for lavaan (its only purpose is to help define the model)
  d_fake <- data.frame(
    x = rnorm(sample_size),
    m = rnorm(sample_size),
    y = rnorm(sample_size)
  )
```

```r
# Build the lavaan model string using glue()
model_string <- glue("
  # Equation for y (direct effect from x is tau_prime)
  y ~ {pop_tau_prime}*x + {pop_beta}*m
  # Equation for m
  m ~ {pop_alpha}*x
  # Fix variances of x, m, and y to 1
  x ~~ 1*x
  y ~~ 1*y
  m ~~ 1*m
")

# Fit the model to extract the implied covariance matrix
fit <- lavaan::lavaan(model = model_string, data = d_fake)
pop_cov <- lavaan::lavInspect(fit, "cor.all")

# Generate a "population" dataset using empirical = TRUE
pop_data <- as.data.frame(
  MASS::mvrnorm(n = sample_size,
                mu = rep(0, 3),
                Sigma = pop_cov,
                empirical = TRUE)
)
pop_rs <- rsquare_med(data = pop_data, x = "x", m = "m", y = "y")

#Run simulation replications (with empirical = FALSE)
sim_matrix <- replicate(num_reps, {
  sim_data <- as.data.frame(
    MASS::mvrnorm(n = sample_size,
                  mu = rep(0, 3),
                  Sigma = pop_cov,
                  empirical = FALSE)
  )
  unlist(rsquare_med(data = sim_data, x = "x", m = "m", y = "y"))
})
sim_means <- rowMeans(sim_matrix)
sim_means["proportionmediated"] <- if (sim_means["total"] != 0) {
  sim_means["mediatedeffect"] / sim_means["total"]
} else NA

# Return a data frame with the random parameters and simulation results
```

```r
    data.frame(
      pop_alpha       = pop_alpha,
      pop_beta        = pop_beta,
      indirect_effect = indirect_effect,
      pop_r2med       = pop_rs$rsquaredmediated,
      sv_r2med        = sim_means["rsquaredmediated"],
      stringsAsFactors = FALSE
    )
  }

  # Set the number of simulations
  n_sim <- 1500

  # Determine the number of cores to use
  n_cores <- detectCores() - 1

  # Create a cluster
  cl <- makeCluster(n_cores)

  # Load required libraries on each worker
  clusterEvalQ(cl, {
    library(MASS)
    library(lavaan)
    library(glue)
  })
```

```
[[1]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[2]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[3]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[4]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"
```

```
[[5]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[6]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[7]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[8]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[9]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[10]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[11]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[12]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[13]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[14]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     

[[15]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"     
```

```
[[16]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[17]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[18]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"

[[19]]
 [1] "glue"      "lavaan"    "MASS"      "stats"     "graphics"  "grDevices"
 [7] "utils"     "datasets"  "methods"   "base"
```

```
# Export required objects to the cluster workers.
# Make sure rsquare_med is defined in your environment or adjust accordingly.
clusterExport(cl, varlist = c("simulate_random_model", "rsquare_med"))

# Run the simulations in parallel using parLapply
random_results_list <- parLapply(cl, 1:n_sim, function(i) {
  simulate_random_model(sample_size = 500, num_reps = 100)
})

# Stop the cluster after finishing the parallel computation
stopCluster(cl)

# Combine the list of data frames into one data frame
random_results <- do.call(rbind, random_results_list)

# Create a scatterplot of the indirect effect vs. the simulated R² mediated.
ggplot(random_results, aes(x = indirect_effect, y = sv_r2med)) +
  geom_point(size = 1, alpha = 1) +
  labs(x = "Indirect Effect (  *  )",
       y = "Simulated R² Mediated (sv_r2med)",
       title = "Simulation with Random   and   (Indirect Effect: [-4, 4], Total Effect = 0.
  scale_x_continuous(limits = c(-4, 4)) +
  scale_y_continuous(limits = c(-1, 1)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
Warning: Removed 9 rows containing missing values or values outside the scale range
(`geom_point()`).
```