R code implementing the examples in A Classical Regression Framework for Mediation Analysis: Fitting One Model to Estimate Mediation Effects

Christina T. Saunders and Jeffrey D. Blume

September 5, 2017

```
### The following R code implements the examples used in the paper
### "A Classical Regression Framework for Mediation Analysis:
### Fitting One Model to Estimate Mediation Effects"
### by Christina T. Saunders and Jeffrey D. Blume
### updated August 5, 2017

### Although the BRAIN-ICU data (as described in the manuscript) cannot be made publically available,
### the following code is used to implement the examples in the manuscript.

###########################################
####### CODE TO IMPLEMENT EXAMPLES ########
###########################################

### Load libraries
library(MASS) # for mvrnorm() function
library(xtable) # for creating tables of results

set.seed(091190) # set seed for reproducibility
nboot <- 10000 # number of bootstrap replications
mcs <- 10000 # number of monte carlo simulations

### function to estimate the indirect effect (and its variance)
### for a unit change in the exposure for continuous x, m, y
### from a *simple mediation model*
### using the Essential Mediation Components (EMCs) formulae,
### Sobel's formula, bootstrapping, and Monte Carlo methods.
### This function is used in Example 1 Part 1 and Example 2.

### NOTE: When (x-x*) = 1, the EMC from the simple mediation model
### is equal to the indirect effect and the portion eliminated.

mediation_fxn <- function(x,m,y){
  #### Fit simple mediation models
  full <-lm(y ~ x + m)
  sub <- lm(y ~ x)
  pathA <- lm(m ~ x)

  # functionals from full model for using formula
  v.xm <- vcov(full)["x","m"]
  v.mx <- vcov(full)["m","x"]
  v.mm <- vcov(full)["m","m"]

  # fitted values and residuals for residual bootstrap
```

```r
f.fit <- full$fitted.values
f.resid <- full$residuals

# extract estimated coefficients
ahat <- pathA$coef["x"]
bhat <- full$coef["m"]
chat <- sub$coef["x"]
cprimehat <- full$coef["x"]

# extract sample variances
svar.c <- vcov(sub)["x","x"]
svar.cprime <- vcov(full)["x","x"]
svar.a <- vcov(pathA)["x","x"]
svar.b <- vcov(full)["m","m"]

# create var-cov matrix for Monte Carlo simulation
# (this specification assumes cov=0)
Sigma.diff <- matrix(c(svar.c,0,0,svar.cprime),byrow=T,ncol=2)
Sigma.prod <- matrix(c(svar.a,0,0,svar.b),byrow=T,ncol=2)

### Formula for the EMC and its variance ####
emc <- v.xm %*% solve(v.mm) * (-1*bhat)
var.emc <- v.xm * solve(v.mm) * v.mx
var.bm <- v.mm

### Calculate IDE using other methods ###
ide.diff <- chat- cprimehat
ide.prod <- ahat*bhat

### Calculate Var(IDE) using delta method approximations
var.sobel <- ahat^2*svar.b + bhat^2*svar.a
var.exact <- ahat^2*svar.b + bhat^2*svar.a + svar.a*svar.b
var.unbiased <- ahat^2*svar.b + bhat^2*svar.a - svar.a*svar.b

############################
######  Bootstrap   ########
############################

## bootstrap cases
boot.ide <- boot.c<- boot.cprime<- boot.a<- boot.b<- rep(NA,nboot)
for(k in 1:nboot){
  # sample with replacement from the rows of (y,x,m)
  dat <- cbind(y,x,m)
  tmp <- sample(1:nrow(dat),nrow(dat),replace=TRUE)
  dat.tmp <- as.data.frame(dat[tmp, ])
  # models
  mf <- lm(y ~ x + m, data=dat.tmp)
  ms <- lm(y ~ x, data=dat.tmp)
  patha <- lm(m ~ x, data=dat.tmp)
  # store results
  boot.ide[k] <- ms$coef["x"] - mf$coef["x"]
  boot.c[k] <- ms$coef["x"]
  boot.cprime[k] <- mf$coef["x"]
  boot.a[k] <- patha$coef["x"]
  boot.b[k] <- mf$coef["m"]
```

```r
}

## bootstrapped covariances of model coefficients from bootstrapping cases
bootcov.diff <- cov(boot.c, boot.cprime)
bootcov.prod <- cov(boot.a, boot.b)
bootrho.diff <- bootcov.diff/(sd(boot.c)*sd(boot.cprime))
bootrho.prod <- bootcov.prod/(sd(boot.a)*sd(boot.b))

#### bootstrap residuals
bootr.ide<- bootr.c<- bootr.cprime<- bootr.a<- bootr.b <- rep(NA,nboot)
for(k in 1:nboot){
  f.resid.dat <- f.resid
  # sample with replacement from the residuals from the full model to get E*.full
  f.tmp <- sample(1:length(f.resid.dat),length(f.resid.dat),replace=TRUE)
  f.resid.dat.tmp <- (f.resid.dat[f.tmp])
  # bootstrapped Y values: Y*.full = Yhat.full + E*.full
  ystar.f <- f.fit + f.resid.dat.tmp

  # regress bootstrapped Y* on fixed design matrix to obtain bootstrap regression coefficients
  mf <- lm(ystar.f ~ x + m)
  ms <- lm(ystar.f ~ x)
  patha <- lm(m~x)
  bootr.ide[k] <- ms$coef["x"] - mf$coef["x"]
  bootr.c[k] <- ms$coef["x"]
  bootr.cprime[k] <- mf$coef["x"]
  bootr.a[k] <- patha$coef["x"]
  bootr.b[k] <- mf$coef["m"]
}

###################
### Monte Carlo ###
###################
Sigma.diff.cov <- matrix(c(svar.c,bootcov.diff,bootcov.diff,svar.cprime),byrow=T,ncol=2)
Sigma.prod.cov <- matrix(c(svar.a,bootcov.prod,bootcov.prod,svar.b),byrow=T,ncol=2)
mc.diff <- mc.prod<- mc.diff.cov<- mc.prod.cov<- rep(NA,mcs)
for(l in 1:mcs){
  # assuming off-diagonals of Sigma are 0
  draw.diff <- mvrnorm(n=1,mu=c(chat,cprimehat),Sigma=Sigma.diff)
  mc.diff[l] <- draw.diff[1]-draw.diff[2]

  draw.prod <- mvrnorm(n=1,mu=c(ahat,bhat),Sigma=Sigma.prod)
  mc.prod[l] <- draw.prod[1]*draw.prod[2]

  ## to use bootstrapped covariances:
  # draw.diff.cov <- mvrnorm(n=1, mu=c(chat,cprimehat), Sigma=Sigma.diff.cov)
  # mc.diff.cov[l] <- draw.diff.cov[1] - draw.diff.cov[2]

  # draw.prod.cov<- mvrnorm(n=1,mu=c(ahat,bhat),Sigma=Sigma.prod.cov)
  # mc.prod.cov[l]<- draw.prod.cov[1]*draw.prod.cov[2]
}

###################
##### 95% CIs #####
###################
```

```r
## function to output confidence intervals
pretty95ci <- function(lower,upper){paste("(",lower,", ",upper,")",sep="")}

##### Sobel Intervals
sobel.ci <- c(ide.prod - qnorm(.975,0,1)*sqrt(var.sobel),
             ide.prod + qnorm(.975,0,1)*sqrt(var.sobel))
sobel.ci.out <- pretty95ci(round(sobel.ci[1],3),round(sobel.ci[2],3))

exact.ci <- c(ide.prod - qnorm(.975,0,1)*sqrt(var.exact),
             ide.prod + qnorm(.975,0,1)*sqrt(var.exact))
exact.ci.out <- pretty95ci(round(exact.ci[1],3), round(exact.ci[2],3))

unbiased.ci <- c(ide.prod - qnorm(.975,0,1)*sqrt(var.unbiased),
                ide.prod + qnorm(.975,0,1)*sqrt(var.unbiased))
unbiased.ci.out <- pretty95ci(round(unbiased.ci[1],3),round(unbiased.ci[2],3))

#### Resampling Intervals
boot.ci <- quantile(boot.ide,c(0.025,0.975))
bootr.ci <- quantile(bootr.ide, c(0.025, 0.975))

#### MC with covariance = 0
mcdiff.ci <- quantile(mc.diff,c(0.025,0.975))
mcprod.ci <- quantile(mc.prod,c(0.025,0.975))

# using EMC formula and t quantiles
p <- 3 # number of parameters (intercept, coefficient for x, coefficient for m)
N <- length(y)
emc.ci <- c(ide.diff + v.xm * solve(v.mm) * qt(.975,df = N-p)*sqrt(var.bm),
           ide.diff - v.xm * solve(v.mm) * qt(.975,df = N-p)*sqrt(var.bm))


#########################
##### Store results #####
#########################

# IDE results
ide.results <- rbind(emc,
                     ide.diff,
                     ide.prod,
                     mean(boot.ide),
                     mean(bootr.ide),
                     mean(mc.prod),
                     mean(mc.diff))
rownames(ide.results) <- c("emc",
                           "ide.diff",
                           "ide.prod",
                           "bootcase.ide",
                           "bootresid.ide",
                           "mc.prod",
                           "mc.diff")
# Varhat(IDEhat) results
var.results <- rbind(var.emc,
                     var.sobel,
                     var.exact,
                     var.unbiased,
                     var(boot.ide),
```

```
                    var(bootr.ide),
                    var(mc.prod),
                    var(mc.diff))
  rownames(var.results) <- c("var.emc",
                             "var.sobel",
                             "var.exact",
                             "var.unbiased",
                             "var.boot.cases",
                             "var.bootresid",
                             "var.mcprod",
                             "var.mcdiff")
  # SE(IDE) results
  sdtab <- round(rbind(sqrt(var.emc),
                       sqrt(var.sobel),
                       sqrt(var.exact),
                       sqrt(var.unbiased),
                       sd(boot.ide),
                       sd(bootr.ide),
                       sd(mc.prod),
                       sd(mc.diff)),2)
  rownames(sdtab) <- c("EMC",
                       "Sobel",
                       "Exact",
                       "Unbiased",
                       "Boot cases",
                       "Boot resid",
                       "MC ab",
                       "MC c-c'")

  # CI results
  citab <- round(rbind("EMC CI:"=emc.ci,
                       "Sobel CI:"=sobel.ci,
                       "Exact CI:"=exact.ci,
                       "Unbiased CI:"=unbiased.ci,
                       "Boot case CI:"= boot.ci,
                       "Boot resid CI:" = bootr.ci,
                       "MC ab:"= mcprod.ci,
                       "MC c-c'"= mcdiff.ci),2)

  list("IDE results" = ide.results,
       "VAR results" = var.results,
       "Boot cases coefs" = list("a" = boot.a,"b" = boot.b,"c" = boot.c,"cprime" = boot.cprime),
       "Boot residual coefs" = list("a" = bootr.a, "b" = bootr.b, "c" = bootr.c, "cprime" = bootr.cprime
       "SDs" = sdtab,
       "CI tab" = citab)
}
```

1 Example 1 (Simple mediation model)

```
#######################################
## Example 1: Simple mediation model ##
#######################################

## load data and assign variables
```

```r
load("example1.Rdata")
y <- s100b # outcome y
m <- sofa # mediator m
x <- bl.cr # predictor x
```

```r
## show structure of data variables for example 1
head(data[,c("bl.cr","sofa","s100b")])

##     bl.cr sofa     s100b
## 175  0.29    2  52.00155
## 178  1.20    8  19.16796
## 179  0.67    4  83.91025
## 182  0.83    6 157.36897
## 185  0.79   10  18.16068
## 187  0.94    9  21.62969
```

```r
## call mediation_fxn() function
example1<- mediation_fxn(x,m,y)

# output results
results <- example1
xtable(results[["IDE results"]],digits=4, caption = "Estimated indirect effect")
```

	x
emc	28.6392
ide.diff	28.6392
ide.prod	28.6392
bootcase.ide	27.8705
bootresid.ide	28.6089
mc.prod	28.6189
mc.diff	28.4952

Table 1: Estimated indirect effect

```r
xtable(results[["SDs"]], caption="Standard errors")
```

	x
EMC	7.12
Sobel	17.25
Exact	17.69
Unbiased	16.80
Boot cases	18.74
Boot resid	7.06
MC ab	17.73
MC c-c'	61.81

Table 2: Standard errors

```r
xtable(results[["VAR results"]],digits=4, caption = "Variance of the estimated indirect effect")
```

	x
var.emc	50.7030
var.sobel	297.5913
var.exact	312.8533
var.unbiased	282.3293
var.boot.cases	351.2407
var.bootresid	49.9081
var.mcprod	314.2489
var.mcdiff	3820.5791

Table 3: Variance of the estimated indirect effect

```
xtable(results[["CI tab"]], caption="Confidence intervals")
```

	x	x
EMC CI:	14.54	42.74
Sobel CI:	-5.17	62.45
Exact CI:	-6.03	63.31
Unbiased CI:	-4.29	61.57
Boot case CI:	-4.17	68.94
Boot resid CI:	15.32	43.42
MC ab:	-2.18	67.35
MC c-c'	-91.51	149.88

Table 4: Confidence intervals

2 Example 1 Part 2

```
#####################################################
## Example 1 part 2: mediation with X and X^2 term ##
#####################################################

xsquared<- x^2
ptm <- proc.time() # start clock
full <- lm(y ~ x + xsquared + m)
v.xm <- vcov(full)[c("x","xsquared"),"m"]
v.m <- vcov(full)["m","m"]
beta.m <- full$coefficients["m"]
(emc <- -1 * v.xm %*% solve(v.m) %*% beta.m) ## EMCs

##            [,1]
## [1,]  28.32743
## [2,] -11.07721

emc[1] + emc[2] ## IDEhat

## [1] 17.25022

(var.emc <- v.xm %*% solve(v.m) %*% t(v.xm)) ## Var(EMCs)

##               x   xsquared
## [1,]  45.02081 -17.605023
## [2,] -17.60502   6.884302
```

```r
sqrt(diag(var.emc)) ## SDs of linear and quadratic components of IDEhat

## [1] 6.709755 2.623795

sqrt(var.emc[1,1] + var.emc[2,2] + 2*var.emc[1,2]) # Var(IDEhat)

##        x
## 4.085959

proc.time() - ptm # Stop the clock

##    user  system elapsed
##   0.016   0.000   0.016

# how to implement with mediate fxn from mediation package by Imai et al.
library(mediation)
library(mgcv)
ptm <- proc.time() # start clock
model.m <- gam(m ~ x + I(x^2))
model.y <- gam(y ~ x + I(x^2) + m)
med <- mediate(model.m = model.m, model.y = model.y, sims=5000,
               treat=c("x","I(x^2)"),
               mediator="m", boot=TRUE)
proc.time() - ptm # Stop the clock

##    user  system elapsed
## 101.801   1.058 102.931

# summary(med)
med$d1 ## IDE

## [1] 17.25022

sd(med$d1.sims) ## SE(IDE)

## [1] 33.26688

## compare total effect from mediate fxn to estimated total effect from fitted model -- they match
med$tau.coef

## [1] 232.8977

lm(y ~ x + I(x^2)) # 81.83 + 151.07 = 232.9

##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Coefficients:
## (Intercept)            x       I(x^2)
##       97.91        81.83       151.07
```

3 Example 2 (Simple mediation model where Sobel's approximation holds)

```r
## show structure of data variables for examples 2,3,4
head(data[, -c(1)])

##          bl.cr bl.epi.mod daily.sofa tot.benz        del
## 1   0.04339465  111.79015          3        0     Normal
## 4   0.93787094  101.14816          5        0 Not normal
## 5   1.69498337   72.91061          7        0     Normal
## 6   1.76925434  108.33077          3        0     Normal
## 7  -0.45087484  112.22816         10        5 Not normal
## 11  0.75720674   95.57906         10        0     Normal
##    rbans.global.score age.enroll charlson.score
## 1                  56   44.99800              0
## 4                  84   50.68775              2
## 5                  82   50.68161              5
## 6                 113   37.25597              0
## 7                  76   59.10596              1
## 11                101   56.46831              3
```

```r
#######################################
## Example 2: Simple mediation model ##
#######################################

# assign variables
x <- bl.cr
m <- daily.sofa
y <- rbans.global.score
```

```r
example2 <- mediation_fxn(x,m,y)

# output results
results<- example2
xtable(results[["IDE results"]],digits=4, caption = "Estimated indirect effect")
```

	x
emc	0.0107
ide.diff	0.0107
ide.prod	0.0107
bootcase.ide	0.0072
bootresid.ide	0.0091
mc.prod	0.0097
mc.diff	0.0109

Table 5: Estimated indirect effect

```r
xtable(results[["SDs"]], caption="Standard errors")
```

```r
xtable(results[["VAR results"]],digits=4, caption = "Variance of the estimated indirect effect")
```

```r
xtable(results[["CI tab"]], caption="Confidence intervals")
```

	x
EMC	0.27
Sobel	0.27
Exact	0.29
Unbiased	0.25
Boot cases	0.28
Boot resid	0.27
MC ab	0.29
MC c-c'	1.94

Table 6: Standard errors

	x
var.emc	0.0736
var.sobel	0.0737
var.exact	0.0823
var.unbiased	0.0650
var.boot.cases	0.0809
var.bootresid	0.0740
var.mcprod	0.0826
var.mcdiff	3.7688

Table 7: Variance of the estimated indirect effect

	x	x
EMC CI:	-0.52	0.55
Sobel CI:	-0.52	0.54
Exact CI:	-0.55	0.57
Unbiased CI:	-0.49	0.51
Boot case CI:	-0.58	0.58
Boot resid CI:	-0.52	0.54
MC ab:	-0.58	0.60
MC c-c'	-3.77	3.91

Table 8: Confidence intervals

4 Example 3 (Exposure-confounder interactions)

```
########################################################################
## Example 3: Model with confounders and exposure-confounder interaction ##
########################################################################

# assign variables
x <- bl.cr # exposure
m <- daily.sofa # mediator
y <- rbans.global.score # outcome
conf <- charlson.score # confounder
```

```
full <- lm(y ~ x + m + conf + x:conf)
m.mod <- lm(m ~ x + conf + x:conf)
sub <- lm(y ~ x + conf + x:conf)

## formula using EMCs
(emc <- -vcov(full)[c("x","x:conf"),"m"] %*% solve(vcov(full)["m","m"]) %*% full$coef["m"])

##              [,1]
## [1,] 2.572539e-03
## [2,] 7.760202e-05

x.treat <- 1 # x
x.ctrl <- 0 # x*
(emc[1] + emc[2]*mean(conf))*(x.treat-x.ctrl) ## portion eliminated, which equals the NIE

## [1] 0.002761358

## mediation formula
(m.mod$coef["x"] + m.mod$coef["x:conf"]*mean(conf))*(full$coef["m"])*(x.treat - x.ctrl)

##           x
## 0.002761358

## compare computation time using EMC formula and mediate function

## using EMC formula...
x.treat <- 1
x.ctrl <- 0
ptm <- proc.time()
full <- lm(y ~ x + m + conf + x:conf)
(emc <- -vcov(full)[c("x","x:conf"),"m"]%*% solve(vcov(full)["m","m"]) %*% full$coef["m"])

##              [,1]
## [1,] 2.572539e-03
## [2,] 7.760202e-05

(ide <- (emc[1] + emc[2]*mean(conf))*(x.treat-x.ctrl))

## [1] 0.002761358

cde <- full$coef["x"] + full$coef["x"]*mean(conf)
sub <- lm(y ~ x + m + conf + x:conf)
te <- sub$coef["x"] + sub$coef["x"]*mean(conf)
# variance
v.emc <- vcov(full)[c("x","x:conf"),"m"]%*% solve(vcov(full)["m","m"]) %*% vcov(full)["m",c("x","x:conf")
v.ide <- (v.emc[1,1]) + (mean(conf)^2 * v.emc[2,2]) + (2*mean(conf)*v.emc[1,2])
sqrt(v.ide) # standard error
```

```
## [1] 0.2208806

proc.time() - ptm # Stop the clock

##    user  system elapsed
##   0.018   0.000   0.018

### look at the 75th percentile of X vs the mean
x.treat <- quantile(x, p=.75)
x.ctrl <- quantile(x, p=.50)
ptm <- proc.time() # start clock
(emc[1] + emc[2]*mean(conf))*(x.treat-x.ctrl)

## Baseline creatinine (mg/dL)
##          75%
## 0.0008156074

proc.time() - ptm

##    user  system elapsed
##   0.002   0.000   0.002
```

```
### repeat using the mediate function...
full <- lm(y ~ x + m + conf + x:conf)
m.mod <- lm(m ~ x + conf + x:conf)
ptm <- proc.time() # start clock
med <- mediate(model.m = m.mod,
               model.y = full,
               treat = "x",
               mediator="m",
               treat.value = 1,
               control.value = 0,
               sims = 5000)
proc.time() - ptm

##    user  system elapsed
##  24.609   0.270  24.892

sd(med$d1.sims[1, ])## SE(IDE)

## [1] 0.2511361

## compare 75th to 50th percentiles of exposure
x.treat <- quantile(x,p=.75)
x.ctrl <- mean(x)

system.time(
  med2 <- mediate(model.m = m.mod,
                  model.y = full,
                  treat = "x",
                  mediator="m",
                  treat.value = x.treat,
                  control.value = x.ctrl,
                  sims = 5000)
)

##    user  system elapsed
##  23.077   0.234  23.321
```

5 Example 4 (Multiple mediator model)

```
#########################################
## Example 4: Multiple mediator model ##
#########################################

## Example with 2 continuous mediators

# assign variables
y <- rbans.global.score
x <- bl.cr
m <- daily.sofa
m2 <- tot.benz
```

```
#############################################
## Single-model approach: fit full model ###
#############################################

full <- lm(y ~ x + m + m2)

## estimate total indirect effect through set of mediators (m,m2)
x.treat <- 1
x.ctrl <- 0
v.xm <- vcov(full)["x",c("m","m2")]
v.mm <- vcov(full)[c("m","m2"), c("m","m2")]
v.mx <- vcov(full)[c("m","m2"), "x"]
beta.m <- c(full$coefficients["m"], full$coefficients["m2"])
(emc <- -1 * v.xm %*% solve(v.mm) %*% beta.m) # EMCs

##             [,1]
## [1,] -0.3410151

total.IDE <- emc %*% (x.treat - x.ctrl) # Total indirect effect
## check that this equals difference of coefficients
# sub<- lm(y ~ x)
# sub£coef["x"] - full£coefficients["x"] ## total IDE

## variance of total IDE
v.xm <- vcov(full)["x",c("m","m2")]
v.mm <- vcov(full)[c("m","m2"), c("m","m2")]
v.mx <- vcov(full)[c("m","m2"), "x"]
var.emc <- v.xm %*% solve(v.mm) %*% v.mx
sqrt(var.emc) # SE(EMC)

##            [,1]
## [1,] 0.3728679

## specific IDE through m
(-1 * vcov(full)["x","m"] * solve(vcov(full)["m","m"]) * full$coefficients["m"])*(x.treat - x.ctrl)

##              [,1]
## [1,] -0.04623919

## check that this equals difference of coefficients
# sub3<- lm(y ~ x + m2) # model excluding m
# sub3£coefficients["x"] - full£coefficients["x"] # IDE of M
```

```
## variance of IDE through m
vcov(full)["x","m"] * solve(vcov(full)["m","m"]) * vcov(full)["m","x"]

##             [,1]
## [1,] 0.09234656

## specific IDE through m2
-1 * vcov(full)["x","m2"] * solve(vcov(full)["m2","m2"]) * full$coefficients["m2"]

##             [,1]
## [1,] -0.3516797

## check that this equals difference of coefficients
# sub2<- lm(y ~ x + m) # model excluding m2
# sub2£coef["x"] - full£coefficients["x"] # IDE of M2

## variance of IDE through m2
vcov(full)["x","m2"] * solve(vcov(full)["m2","m2"]) * vcov(full)["m2","x"]

##             [,1]
## [1,] 0.06568508
```

```
###################################################
## MacKinnon's parallel (aka Hayes' single-step) ##
## and VanderWeele's regression-based approach   ##
###################################################

## fit full model and a separate model for each mediator
full<- lm(y ~ x + m + m2)
pathA1<- lm(m ~ x)
pathA2<- lm(m2 ~ x)
# IDE through m estimated using ...
(a1b1<- pathA1$coefficients["x"]*full$coefficients["m"]) # IDE through m

##           x
## -0.04161422

(a2b2<- pathA2$coefficients["x"]*full$coefficients["m2"]) # IDE through m2

##          x
## -0.2994008

a1b1 + a2b2 ## these sum to total IDE

##          x
## -0.3410151

# extract model coefficients
a1<- pathA1$coefficients["x"]
a2<- pathA2$coefficients["x"]
b1<- full$coefficients["m"]
b2<- full$coefficients["m2"]
# extract sample variances of coefficients
s2_a1<- vcov(pathA1)["x","x"]
s2_a2<- vcov(pathA2)["x","x"]
s2_b1<- vcov(full)["m","m"]
```

```r
s2_b2<- vcov(full)["m2","m2"]
sb1b2<- vcov(full)["m","m2"] # cov between b1 and b2

## variance of IDE through M1: a1^2 * s2_b1 + b1^2 * s2_a1 (from MacKinnon's book pg 107)
(a1^2 * s2_b1) + (b1^2 * s2_a1)

##          x
## 0.07500048

## variance of IDE through M2: a2^2 * s2_b2 + b2^2 * s2_a2
(a2^2 * s2_b2) + (b2^2 * s2_a2)

##          x
## 0.06416167

## variance of total IDE through set M1 and M2:
(s2_a1*b1^2) + (s2_b1*a1^2) + (s2_a2 * b2^2) + (s2_b2 * a2^2) + (2*a1*a2*sb1b2)

##         m
## 0.1557879
```

```r
###############################################
## Hayes' serial model / sequential approach ##
###############################################

# assuming X --> M --> M2 --> Y
full<- lm(y ~ x + m + m2)
# m2 depends on M and X
sub1<- lm(m2 ~ x + m)
# M depends on X
sub2<- lm(m ~ x)

## IDE of X through M to Y =  alpha1*beta1
alpha1<- sub2$coefficients["x"]
s2_alpha1<- vcov(sub2)["x","x"]

beta1<- full$coefficients["m"]
s2_beta1<- vcov(full)["m","m"]

alpha1*beta1

##           x
## -0.04161422

## IDE of X through M2 to Y = alpha2*beta2
alpha2<- sub1$coef["x"]
s2_alpha2<- vcov(sub1)["x","x"]

beta2<- full$coefficients["m2"]
s2_beta2<- vcov(full)["m2","m2"]

alpha2*beta2

##          x
## -0.3516797
```

```
## IDE of X through M1 to M2 to Y = alpha_1*d21*beta2
d21<- sub1$coefficients["m"]
s2_d21 <- vcov(sub1)["m","m"]
alpha1 * d21 * beta2

##          x
## 0.05227883

(alpha1*beta1) +  (alpha2*beta2) + (alpha1 * d21 * beta2) ## these sum to total IDE

##          x
## -0.3410151

## variance of alpha1*beta1 is a1^2 * s2_b1 + b1^2 * s2_a1
v_a1b1 <- (alpha1^2 * s2_beta1) + (beta1^2 * s2_alpha1)
v_a1b1

##          x
## 0.07500048

(alpha1*beta1) + c(-1,1)*qnorm(.975)*sqrt(v_a1b1)

## [1] -0.5783742  0.4951458

## variance of alpha2*beta2 is a2^2 * s2_b2 + b2^2 * s2_a2
v_a2b2 <- (alpha2^2 * s2_beta2) + (beta2^2 * s2_alpha2)
sqrt(v_a2b2)

##         x
## 0.2874698

alpha2*beta2 + c(-1,1)*qnorm(.975)*sqrt(v_a2b2)

## [1] -0.9151101  0.2117508

## variance of alpha1*d21*beta2:
v_a1_d21_b2<- (alpha1^2 * d21^2 * s2_beta2) + (alpha1^2 * beta2^2 * s2_d21) + (d21^2 * beta2^2 + s2_alpha
sqrt(v_a1_d21_b2)

##         x
## 0.3559383

alpha1*d21*beta2 + c(-1,1)*qnorm(.975)*sqrt(v_a1_d21_b2)

## [1] -0.6453473  0.7499050

## switch temporal order of mediators: X --> M2 --> M --> Y
full<- lm(y ~ x + m2 + m)
# M depends on M2 and X
sub1<- lm(m ~ x + m2)
# M2 depends on X
sub2<- lm(m2 ~ x)

## IDE of X through M2 to Y: alpha1*beta1
sub2$coef["x"]*full$coefficients["m2"]

##          x
## -0.2994008
```

```r
## IDE of X through M to Y: alpha2*beta2
sub1$coef["x"]*full$coefficients["m"]

##           x
## -0.04623919

## IDE of X through M2 to M1 to Y: alpha_1 *d21 *Beta2
sub2$coefficients["x"] * sub1$coef["m2"] * full$coefficients["m"]

##           x
## 0.004624971

# these specific IDEs again sum to total IDE
```