# EVR SIMULATION CODE FOR CONTINUOUS MODERATOR

# Michael Smithson, The Australian National University

#

# For moderator effect on regression coefficient only set dr1 = 0

# and dy1 > 0.

#

# Load the relevant libraries:

library(MASS)

library(mvtnorm)

#

# Regression moderator data generating function:

datagen1 <- function(n,dx0,dx1,dy0,dy1,dr0,dr1) {

z <- rnorm(n,mean = 0, sd = 1);

sx <- exp(dx0+dx1\*z);

sy <- exp(dy0+dy1\*z);

rho <- (exp(dr0+dr1\*z) - 1)/(exp(dr0+dr1\*z) + 1)

mat <- rep(c(1:n),4)

dim(mat) = c(2,2,n)

for (i in 1:n) mat[,,i] <- matrix(c(sx[i]^2,rho[i]\*sx[i]\*sy[i], rho[i]\*sx[i]\*sy[i],sy[i]^2),ncol = 2)

vec1 <- rep(c(1:n),2)

dim(vec1) <- c(n,2)

for (i in 1:n) vec1[i,] <- rmvnorm(1,mean = c(0,0), sigma = mat[,,i], method = "svd")

cbind(vec1,z)

}

#

# For moderator effect on correlation only, use datagen2 function:

datagen2 <- function(n,dx0,dx1,dy0,dy1,dr0) {

z <- rnorm(n,mean = 0, sd = 1);

sx <- exp(dx0+dx1\*z);

sy <- exp(dy0+dy1\*z);

rho <- exp(dr0)\*sx/sy;

mat <- rep(c(1:n),4)

dim(mat) = c(2,2,n)

for (i in 1:n) mat[,,i] <- matrix(c(sx[i]^2,rho[i]\*sx[i]\*sy[i], rho[i]\*sx[i]\*sy[i],sy[i]^2),ncol = 2)

vec1 <- rep(c(1:n),2)

dim(vec1) <- c(n,2)

for (i in 1:n) vec1[i,] <- rmvnorm(1,mean = c(0,0), sigma = mat[,,i], method = "svd")

cbind(vec1,z)

}

#

# Model for dy = dx:

hevrho0 <- function(h, y, x, zx,zr)

{

L1 <- length(zx[1,])+1; L3 <- L1+length(zr[1,])-1; L4 <- L3+1; L5 <- L3+2;

mux <- h[L4]; muy <- h[L5];

# The output shows dx, dy, dr, mux, muy in that order.

devx <- x - mux

devy <- y - muy

dx = zx%\*%h[1:length(zx[1,])]

dr = zr%\*%h[L1:L3]

sx <- exp(dx)

sy <- exp(dx)

rho <- (exp(dr)-1)/(exp(dr)+1)

loglik = ((sx^2)\*(devy^2) - 2\*rho\*sx\*sy\*devx\*devy + (sy^2)\*(devx^2))/(2\*(rho^2-1)\*(sx^2)\*(sy^2)) - 0.5\*(log(1-rho^2) + log(sx^2) + log(sy^2))

-sum(loglik, na.rm = TRUE)

}

#

# Model for dy not equal to dx:

hevrho1 <- function(h, y, x, zx,zy,zr)

{

L1 <- length(zx[1,])+1; L2 <- L1+length(zy[1,])-1; L21 <- L2+1; L3 <- L21+length(zr[1,])-1; L4 <- L3+1; L5 <- L3+2;

mux <- h[L4]; muy <- h[L5];

# The output shows dx, dy, dr, mux, muy in that order.

devx <- x - mux

devy <- y - muy

dx = zx%\*%h[1:length(zx[1,])]

dy = zy%\*%h[L1:L2]

dr = zr%\*%h[L21:L3]

sx <- exp(dx)

sy <- exp(dy)

rho <- (exp(dr)-1)/(exp(dr)+1)

loglik = ((sx^2)\*(devy^2) - 2\*rho\*sx\*sy\*devx\*devy + (sy^2)\*(devx^2))/(2\*(rho^2-1)\*(sx^2)\*(sy^2)) - 0.5\*(log(1-rho^2) + log(sx^2) + log(sy^2))

-sum(loglik, na.rm = TRUE)

}

#

# Model generating and comparison function for regression models:

modgen1 <- function(n,dx0,dx1,dy0,dy1,dr0,dr1) {

tdata <- as.data.frame(datagen1(n,dx0,dx1,dy0,dy1,dr0,dr1));

ydata <- tdata[,2];

const <- rep(1,length(ydata));

xdata <- tdata[,1];

# Model 0: dx1 = dy1 and dr1 = 0

zxdata <- cbind(const, tdata[,3]); zrdata <- cbind(const);

start <- c(dx0,dx1,dr0,0,0);

hevreg0 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg0$par

hevreg0 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata, method = "BFGS");

# Model 1: dx1 != dy1 and dr1 = 0

zxdata <- cbind(const, tdata[,3]); zydata <- cbind(const, tdata[,3]); zrdata <- cbind(const);

start <- c(dx0,dx1,dy0,dy1,dr0,0,0);

hevreg1 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg1$par

hevreg1 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata, method = "BFGS");

# Model 2: dx1 = dy1 and dr1 != 0

zxdata <- cbind(const, tdata[,3]); zrdata <- cbind(const, tdata[,3]);

start <- c(dx0,dx1,dr0,dr1,0,0);

hevreg2 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg2$par

hevreg2 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata, method = "BFGS");

# Model 3: dx1 != dy1 and dr1 != 0

zxdata <- cbind(const, tdata[,3]); zydata <- cbind(const, tdata[,3]); zrdata <- cbind(const, tdata[,3]);

start <- c(dx0,dx1,dy0,dy1,dr0,dr1,0,0);

hevreg3 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg3$par

hevreg3 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata, method = "BFGS");

#

loglike <- c(hevreg0$value, hevreg1$value, hevreg2$value, hevreg3$value);

# Compare mod0 vs mod1, mod0 vs mod2, mod1 vs mod3, mod2 vs mod3:

tvec <- c(1-pchisq(2\*(loglike[1]-loglike[2]),2), 1-pchisq(2\*(loglike[1]-loglike[3]),1), 1-pchisq(2\*(loglike[2]-loglike[4]),1), 1-pchisq(2\*(loglike[3]-loglike[4]),2));

tvec

}

#

# Model generating and comparison function for correlation models:

modgen2 <- function(n,dx0,dx1,dy0,dy1,dr0) {

tdata <- as.data.frame(datagen2(n,dx0,dx1,dy0,dy1,dr0));

ydata <- tdata[,2];

const <- rep(1,length(ydata));

xdata <- tdata[,1];

# Model 0: dx1 = dy1 and dr1 = 0

zxdata <- cbind(const, tdata[,3]); zrdata <- cbind(const);

start <- c(dx0,dx1,dr0,0,0);

hevreg0 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg0$par

hevreg0 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata, method = "BFGS");

# Model 1: dx1 != dy1 and dr1 = 0

zxdata <- cbind(const, tdata[,3]); zydata <- cbind(const, tdata[,3]); zrdata <- cbind(const);

start <- c(dx0,dx1,dy0,dy1,dr0,0,0);

hevreg1 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg1$par

hevreg1 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata, method = "BFGS");

# Model 2: dx1 = dy1 and dr1 != 0

zxdata <- cbind(const, tdata[,3]); zrdata <- cbind(const, tdata[,3]);

start <- c(dx0,dx1,dr0,0,0,0);

hevreg2 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg2$par

hevreg2 <- optim(start, hevrho0, hessian = T, x = xdata, y = ydata, zx = zxdata, zr = zrdata, method = "BFGS");

# Model 3: dx1 != dy1 and dr1 != 0

zxdata <- cbind(const, tdata[,3]); zydata <- cbind(const, tdata[,3]); zrdata <- cbind(const, tdata[,3]);

start <- c(dx0,dx1,dy0,dy1,dr0,0,0,0);

hevreg3 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata);

# Now use BFGS to improve the estimate, using the previous model

# parameters as starting values.

start <- hevreg3$par

hevreg3 <- optim(start, hevrho1, hessian = T, x = xdata, y = ydata, zx = zxdata, zy = zydata, zr = zrdata, method = "BFGS");

#

loglike <- c(hevreg0$value, hevreg1$value, hevreg2$value, hevreg3$value);

# Compare mod0 vs mod1, mod0 vs mod2, mod1 vs mod3, mod2 vs mod3:

tvec <- c(1-pchisq(2\*(loglike[1]-loglike[2]),2), 1-pchisq(2\*(loglike[1]-loglike[3]),1), 1-pchisq(2\*(loglike[2]-loglike[4]),1), 1-pchisq(2\*(loglike[3]-loglike[4]),2));

tvec

}

#

# Loop modgen1 many (sim) times:

modsim1 <- function(sim,n,dx0,dx1,dy0,dy1,dr0,dr1) {

outmat <- matrix(c(rep(0,4\*sim)),ncol = 4)

for (i in 1:sim) {

treg <- modgen1(n,dx0,dx1,dy0,dy1,dr0,dr1)

outmat[i,1] <- ifelse(treg[1] <= .05,1,0)

outmat[i,2] <- ifelse(treg[2] <= .05,1,0)

outmat[i,3] <- ifelse(treg[3] <= .05,1,0)

outmat[i,4] <- ifelse(treg[4] <= .05,1,0)

}

reject <- rbind(colSums(outmat),c(rep(sim,4)),c(n,NA,NA,NA))

colnames(reject) <- c("dx1-dy1","dr1","dr1","dx1-dy1")

rownames(reject) <- c("rejections","runs","samples")

reject

}

#

# Loop modgen2 many (sim) times:

modsim2 <- function(sim,n,dx0,dx1,dy0,dy1,dr0) {

outmat <- matrix(c(rep(0,4\*sim)),ncol = 4)

for (i in 1:sim) {

treg <- modgen2(n,dx0,dx1,dy0,dy1,dr0)

outmat[i,1] <- ifelse(treg[1] <= .05,1,0)

outmat[i,2] <- ifelse(treg[2] <= .05,1,0)

outmat[i,3] <- ifelse(treg[3] <= .05,1,0)

outmat[i,4] <- ifelse(treg[4] <= .05,1,0)

}

reject <- rbind(colSums(outmat),c(rep(sim,4)),c(n,NA,NA,NA))

colnames(reject) <- c("dx1-dy1","dr1","dr1","dx1-dy1")

rownames(reject) <- c("rejections","runs","samples")

reject

}

#

# Example of a 20,000-run simulaton with sample size 70, and

# dx0,dx1,dy0,dy1,dr0,dr1 all = 0:

> T70n000000 <- modsim1(20000,70, 0,0,0,0,0,0); T70n000000