# EVR SIMULATION CODE FOR CATEGORICAL MODERATOR

# Michael Smithson, The Australian National University

#

# Load the requisite libraries:

library(mvtnorm)

library(MASS)

#

# Generate the data:

gendata <- function(n1,n2,mean1,mean2,sigma1,sigma2) {

mat1 <- rmvnorm(n1, mean = mean1, sigma = sigma1)

mat2 <- rmvnorm(n2, mean = mean2, sigma = sigma2)

z <- c(rep(-1,n1),rep(1,n2))

mattot <- cbind(rbind(mat1,mat2),z)

colnames(mattot) <- c("x","y","z")

mattot

}

#

# Method of moment estimation function for starting values:

MMest <- function(data,x,y,z) {

mnx <- mean(x)

mny <- mean(y)

zed <- factor(z)

sdx <- tapply(x,zed,sd)

sdy <- tapply(y,zed,sd)

dx0 <- (log(sdx[[1]]) + log(sdx[[2]]))/2

dx1 <- dx0 - log(sdx[[1]])

dy0 <- (log(sdy[[1]]) + log(sdy[[2]]))/2

dy1 <- dy0 - log(sdy[[1]])

ymod1 <- lm(y~x,subset(data,z==-1))

cor1 <- ymod1$coeff[[2]]\*sdx[[1]]/sdy[[1]]

ymod2 <- lm(y~x,subset(data,z==1))

cor2 <- ymod2$coeff[[2]]\*sdx[[2]]/sdy[[2]]

logit1 <- log((cor1+1)/(1-cor1))

logit2 <- log((cor2+1)/(1-cor2))

dr0 <- (logit1 + logit2)/2

dr1 <- dr0 - logit1

params <- c(dx0,dx1,dy0,dy1,dr0,dr1,mnx,mny)

params

}

#

# MLE for moderator model:

#The output is in this order: dx0, dx1, dy0, dy1, dr0, dr1, muX, muY, dxy1

# The last estimate is dxy1 = dx1 – dy1

#

hevmod <- 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]

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)) - log(2\*pi)- 0.5\*(log(1-rho^2) + log(sx^2) + log(sy^2))

-sum(loglik, na.rm = TRUE)

}

#

# Generate the model:

modgen <- function(n1,n2,mean1,mean2,sigma1,sigma2) {

tdata <- as.data.frame(gendata(n1,n2,mean1,mean2,sigma1,sigma2));

ydata <- cbind(tdata$y);

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

xdata <- cbind(tdata$x);

zxdata <- cbind(const,tdata$z); zydata <- cbind(const,tdata$z); zrdata <- cbind(const,tdata$z);

start <- MMest(tdata,tdata$x,tdata$y,tdata$z)

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

outmod <- rbind(estim <- hevreg$par, serr <- sqrt(diag(solve(hevreg$hessian))), zstat <- estim/serr, prob <- 2\*(1-pnorm(abs(zstat)))); row.names(outmod) <- c("estim", "serr", "zstat", "prob");

vcovpar <- solve(hevreg$hessian);

dxy1 <- outmod[1,2]-outmod[1,4];

dxy1se <- sqrt(vcovpar[2,2] + vcovpar[4,4] - 2\*vcovpar[2,4]);

zdxy1 <- dxy1/dxy1se;

pdxy1 <- 1-pnorm(abs(zdxy1),0,1);

totout <- cbind(outmod, c(dxy1,dxy1se,zdxy1,pdxy1));

totout

}

#

#

# Now loop it many (sim) times:

modsim <- function(sim,n1,n2,mean1,mean2,sigma1,sigma2) {

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

for (i in 1:sim) {

treg <- modgen(n1,n2,mean1,mean2,sigma1,sigma2)

outmat[i,1] <- ifelse(abs(treg[3,2]) >= qt(.975,n1+n2-2),1,0)

outmat[i,2] <- ifelse(abs(treg[3,4]) >= qt(.975,n1+n2-2),1,0)

outmat[i,3] <- ifelse(abs(treg[3,6]) >= qt(.975,n1+n2-2),1,0)

outmat[i,4] <- ifelse(abs(treg[3,9]) >= qt(.975,n1+n2-2),1,0)

 }

reject <- rbind(colSums(outmat),c(rep(sim,4)),c(n1,n2,NA,NA))

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

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

reject

}

#

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

# variance-covariance matrices =

# matrix(c(2,1,1,2),ncol = 2),

# matrix(c(2,sqrt(3/2),sqrt(3/2),3),ncol = 2):

> T570701 <- modsim(20000,70,70,c(0,0),c(0,0), matrix(c(2,1,1,2),ncol = 2), matrix(c(2,sqrt(3/2),sqrt(3/2),3),ncol = 2)); T570701