# Appendix A: Hypothetical Individual Patient Data

|  |  |  |
| --- | --- | --- |
| **Patient number** | **U1** | **U2** |
| 1 | 0.736 | 0.700 |
| 2 | 0.698 | 0.625 |
| 3 | 0.756 | 0.719 |
| 4 | 0.638 | 0.594 |
| 5 | 0.646 | 0.565 |
| 6 | 0.619 | 0.565 |
| 7 | 0.800 | 0.779 |
| 8 | 0.411 | 0.363 |
| 9 | 0.664 | 0.540 |
| 10 | 0.513 | 0.500 |
| 11 | 0.594 | 0.632 |
| 12 | 0.371 | 0.300 |
| 13 | 0.608 | 0.572 |
| 14 | 0.528 | 0.420 |
| 15 | 0.823 | 0.810 |
| 16 | 0.682 | 0.650 |
| 17 | 0.462 | 0.457 |
| 18 | 0.576 | 0.517 |
| 19 | 0.572 | 0.495 |
| 20 | 0.607 | 0.561 |
| 21 | 0.542 | 0.481 |
| 22 | 0.625 | 0.610 |
| 23 | 0.694 | 0.603 |
| 24 | 0.512 | 0.463 |
| 25 | 0.591 | 0.551 |
| 26 | 0.559 | 0.450 |
| 27 | 0.588 | 0.550 |
| 28 | 0.766 | 0.647 |
| 29 | 0.551 | 0.491 |
| 30 | 0.256 | 0.150 |

Table A1: HRQL values for 30 hypothetical patients with a disease in either a less severe (U1) or more severe (U2) state

# Appendix B: R Code for implementing all methods and analyses

rm(list=ls())

setwd("X:/Manuscripts/03 To Submit/Monotonicity/Manuscript for Submission/Figures/")

#load("MattData.rData")

require(MASS)

Data.2D <- data.frame(

U1=

c(0.736, 0.698, 0.756, 0.638, 0.646,

0.619, 0.800, 0.411, 0.664, 0.513,

0.594, 0.371, 0.608, 0.528, 0.823,

0.682, 0.462, 0.576, 0.572, 0.607,

0.542, 0.625, 0.694, 0.512, 0.591,

0.559, 0.588, 0.766, 0.551, 0.256

),

U2=

c(0.700, 0.625, 0.719, 0.594, 0.565,

0.565, 0.779, 0.363, 0.540, 0.500,

0.532, 0.300, 0.572, 0.420, 0.810,

0.650, 0.457, 0.517, 0.495, 0.561,

0.481, 0.610, 0.603, 0.463, 0.551,

0.450, 0.550, 0.647, 0.491, 0.150

)

)

mean(Data.2D$U1);mean(Data.2D$U2)

N.U1 <- length(Data.2D$U1)

N.U2 <- length(Data.2D$U2)

#find 95% CI for U1 and U2

u1.low<-mean(Data.2D$U1)-1.96\*sd(Data.2D$U1)/sqrt(N.U1)

u1.up<-mean(Data.2D$U1)+1.96\*sd(Data.2D$U1)/sqrt(N.U2)

u2.low<-mean(Data.2D$U2)-1.96\*sd(Data.2D$U2)/sqrt(N.U1)

u2.up<-mean(Data.2D$U2)+1.96\*sd(Data.2D$U2)/sqrt(N.U2)

u1.low;u1.up;u2.low;u2.up

# reproduce figure 1

tiff("Fig1.tiff", 400, 400)

plot(U2 ~ U1, data=Data.2D, xlim=c(0,1), ylim=c(0,1), xlab="Utility in moderate state (U1)", ylab="Utility in severe state (U2)")

abline(0,1)

dev.off()

# What is the true variance-covariance of the datasets?

cov(Data.2D)

cor(Data.2D)

# Quick example to show why methods 3 and 4 will produce biased means.

#norm.untruncated <- rnorm(1000, 0, 1)

#norm.truncated <- norm.untruncated

#norm.truncated[norm.untruncated < -0.5] <- -0.5

#cov(Data.3D)

#cor(Data.3D)

# The data themselves. This is the only information available to the

# ten methods, in addition to direction of the monotonic relationship:

# Note: the sds are standard deivations of the means - i.e. standard errors

U1.summary <- list(mu=round(mean(Data.2D$U1),3), sd=(round(u1.up,3) - round(mean(Data.2D$U1),3))/1.96)

U2.summary <- list(mu=round(mean(Data.2D$U2),3), sd=(round(u2.up,3) - round(mean(Data.2D$U2),3))/1.96)

############################################################################################################################

######################## METHODS ###########################################################################################

############################################################################################################################

## Method 1 : Independent Sampling (Naive)

## Method 2 : Quantile matching/same random number seed

## Method 3 : Upward Replacement

## Method 4 : Downward Replacement

## Method 5 : Upward Resampling

## Method 6 : Downward Resampling

## Method 7 : AIVM Covariance

## Method 8 : Lower bounded covariance retrofitting

## Method 9 : Upper Bounded covariance retrofitting

## Method 10: Beta distribution difference modelling

############################################################################################################################

n.PSA <- 1000

# Bootstrapping for comparison

methodBoot.PSA <- matrix(NA, ncol=2, nrow=n.PSA)

for (i in 1:n.PSA){

draws <- 1: dim(Data.2D)[1]

size=dim(Data.2D)[1]

tmp <- Data.2D[sample(draws, size, T),]

methodBoot.PSA[i,] <- c(mean(tmp[,1]), mean(tmp[,2]))

}

methodBoot.PSA <- data.frame(methodBoot.PSA)

names(methodBoot.PSA) <- c("u1","u2")

tiff("Fig2 PSA\_boot.tiff",400, 400)

plot(u2 ~ u1, data=methodBoot.PSA, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="")

abline(0,1)

dev.off()

estBeta <- function(mu, var) {

a <- mu \* ((1 - mu) \* (mu / var) - 1)

b <- a \* ((1 - mu) / mu)

return(list(a=a, b=b))

}

# METHOD 1: INDEPENDENT SAMPLING

u1.param <- estBeta(U1.summary$mu, U1.summary$sd^2)

u2.param <- estBeta(U2.summary$mu, U2.summary$sd^2)

PSA.method01 <- data.frame(u1=rbeta(n.PSA,

u1.param$a,

u1.param$b

),

u2=rbeta(n.PSA,

u2.param$a,

u2.param$b

)

)

plot(u2 ~ u1, data=PSA.method01, xlim=c(0.45, 0.7), ylim=c(0.45,0.7))

abline(0,1)

# METHOD 2: SAME RANDOM NUMBER SEED

# Illustration of issue with random number stream and beta distribution

# non-problematic run:

set.seed(80)

u1 <- rbeta(n.PSA,

u1.param$a,

u1.param$b

)

set.seed(80)

u2 <- rbeta(n.PSA,

u2.param$a,

u2.param$b

)

PSA.method02 <- data.frame(u1=u1, u2=u2)

rm(u1, u2)

plot(u2 ~ u1, data=PSA.method02, xlim=c(0.45, 0.7), ylim=c(0.45,0.7))

abline(0,1)

# METHOD 3: UPWARD REPLACEMENT

u1 <- rbeta(n.PSA, u1.param$a, u1.param$b)

u2 <- rbeta(n.PSA, u2.param$a, u2.param$b)

u1[u1 < u2] <- u2[u1 < u2]

PSA.method03 <- data.frame(u1=u1, u2=u2)

rm(u1, u2)

plot(u2 ~ u1, data=PSA.method03, xlim=c(0.45, 0.7), ylim=c(0.45,0.7))

abline(0,1)

# METHOD 4: DOWNWARD REPLACEMENT

u1 <- rbeta(n.PSA, u1.param$a, u1.param$b)

u2 <- rbeta(n.PSA, u2.param$a, u2.param$b)

u2[u2 > u1] <- u1[u2 > u1]

PSA.method04 <- data.frame(u1=u1, u2=u2)

rm(u1, u2)

plot(u2 ~ u1, data=PSA.method04, xlim=c(0.45, 0.7), ylim=c(0.45,0.7))

abline(0,1)

# METHOD 5: UPWARDS RESAMPLING

u1 <- rbeta(n.PSA, u1.param$a, u1.param$b)

u2 <- rep(NA, n.PSA)

for (i in 1:n.PSA){

continue <- F

while(continue==F){

this.u2 <- rbeta(1, u2.param$a, u2.param$b)

if (this.u2 < u1[i]){

u2[i] <- this.u2

continue <- T

}

}

}

PSA.method05 <- data.frame(u1=u1, u2=u2)

rm(u1, u2)

plot(u2 ~ u1, data=PSA.method05, xlim=c(0.45, 0.7), ylim=c(0.45,0.7))

abline(0,1)

# METHOD 6: DOWNWARDS RESAMPLING [?]

u1 <- rep(NA, n.PSA)

u2 <- rbeta(n.PSA, u2.param$a, u2.param$b)

for (i in 1:n.PSA){

continue <- F

while(continue==F){

this.u1 <- rbeta(1, u1.param$a, u1.param$b)

if (this.u1 > u2[i]){

u1[i] <- this.u1

continue <- T

}

}

}

PSA.method06 <- data.frame(u1=u1, u2=u2)

rm(u1, u2)

# METHOD 7: AIVM COVARIANCE

# Correlation(X, Y) := covariance (X, Y) / (sd(X) \* sd(Y))

# So, when correlation = 1

# covariance(X, y) = sd(X) \* sd(Y)

# This defines the upper limit on the values

# Function

require(MASS)

MakeAIVMCov.2d <- function(mu.X, sd.X, mu.Y, sd.Y, n.psa=n.PSA){

varX <- sd.X^2

varY <- sd.Y^2

aivm <- min(

mean(

c(varX, varY)

),

sd.X \* sd.Y)

sig <- matrix(data=c(varX, aivm, aivm, varY), nrow=2, byrow=T)

aivm.samples <- mvrnorm(n=n.psa, mu=c(mu.X, mu.Y), Sigma=sig )

colnames(aivm.samples) <- c("X.sampled", "Y.sampled")

aivm.samples <- as.data.frame(aivm.samples)

return(list(aivm.samples=aivm.samples, aivm=aivm))

}

tmp <- MakeAIVMCov.2d(

mu.X=U1.summary$mu,

sd.X=U1.summary$sd,

mu.Y=U2.summary$mu,

sd.Y=U2.summary$sd

)

PSA.method07 <- tmp$aivm.samples

names(PSA.method07) <- c("u1", "u2")

#plot(u2 ~ u1, data=PSA.method07)

# METHOD 8: Lower Bounded Covariance Retrofitting

# METHOD 9: Upper Bounded Covariance Retrofitting

# Lowerbounded (method 8) : use upper=F

# Otherwise (method 9) use default: upper=T

MakeBCVR.2d <- function(mu.X, sd.X, mu.Y, sd.Y, n.psa=n.PSA, incBy=0.00001, upper=T){

require(MASS)

varX <- sd.X^2 # variance of X

varY <- sd.Y^2 # variance of Y

if(upper==T){

lowerbound <- 0 # start assuming independent

upperbound <- min(sd.X \* sd.Y,

mean(varX, varY)

) # upper bounds are the minimum of the AIVM or the cov which implies a cor > 1

} else {

lowerbound <- mean(varX, varY)

upperbound <- sd.X \* sd.Y # don't select a covariance which implies a correlation > 1

}

this.cov <- lowerbound

cat(varX, varY, lowerbound, upperbound, this.cov, "\n")

mus <- c(mu.X, mu.Y)

search <- T

if(this.cov==upperbound){ # if the maximum value's been reached already

cat("Upperbound already reached\n")

search <- F # if the upper limit's already been reached, go no further

testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)

testsamples <- mvrnorm(n.psa, mu=mus, Sigma=testsig)

} else {

cat("Upperbound not yet reached\n")

this.cov <- lowerbound

cat("This covariance: ", this.cov, "\n", sep="")

testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)

testsamples <- mvrnorm(n.psa, mu=mus, Sigma=testsig)

}

while(search==T){

cat("trying ", this.cov, "\n")

testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)

try.testsamples <- try(mvrnorm(n.psa, mu=mus, Sigma=testsig))

if(class(try.testsamples)=="try-error"){ # if mvrnorm has been passed impossible values

search <- F

cat("Error picked up\n")

} else {

cat("No error in mvrnorm args\n")

testsamples <- try.testsamples # if the attempted values are correct, use them

if (any(testsamples[,1] < testsamples[,2])){

cat("Violation with ", this.cov, "\n")

this.cov <- this.cov + incBy # increment the values by a little bit

cat("Trying ", this.cov, "\n")

} else {

cat("Found ", this.cov, "\n")

search <- F

}

}

}

this.cor <- this.cov / (sd.X \* sd.Y)

return(list(cov=this.cov, samples=testsamples, cor=this.cor))

}

tmp <- MakeBCVR.2d(

mu.X=U1.summary$mu,

sd.X=U1.summary$sd,

mu.Y=U2.summary$mu,

sd.Y=U2.summary$sd,

upper=F

)

method08.cov <- tmp$cov

method08.cor <- tmp$cor

PSA.method08 <-data.frame(tmp$samples)

names(PSA.method08) <- c("u1", "u2")

tmp <- MakeBCVR.2d(

mu.X=U1.summary$mu,

sd.X=U1.summary$sd,

mu.Y=U2.summary$mu,

sd.Y=U2.summary$sd

)

method09.cov <- tmp$cov

method09.cor <- tmp$cor

PSA.method09 <- data.frame(tmp$samples)

names(PSA.method09) <- c("u1", "u2")

plot(u2 ~ u1, data=PSA.method08)

plot(u2 ~ u1, data=PSA.method09)

# METHOD 10: Beta distribution difference fitting

getDifParam <- function(u1.mu, u1.sd, u2.mu, u2.sd, quietly=T){

mu <- u1.mu - u2.mu

sigma2 <- ifelse(u1.sd > u2.sd, u1.sd^2 - u2.sd^2, u2.sd^2 - u1.sd^2)

x <- (1 - mu) / mu

a <- (x/sigma2-1-2\*x-x^2)/(1+3\*x+3\*x^2+x^3)

b<-a\*x

if(quietly==F){

print(a/(a+b)) # check mean of delta

print(a\*b/(a+b)^2/(a+b+1)) # check variance of delta

}

return(list(a=a, b=b))

}

DifParams <- getDifParam(U1.summary$mu, U1.summary$sd, U2.summary$mu, U2.summary$sd)

n.PSA <- 1000

#rU2.raw <- rnorm(n.PSA, U2.summary$mu, U2.summary$sd)

rU2.raw <- rbeta(n.PSA, u2.param$a, u2.param$b)

#rU1 <- rnorm(n.PSA, U1.summary$mu, U1.summary$sd)

rU1 <- rbeta(n.PSA, u1.param$a, u1.param$b)

rdelta<-rbeta(n.PSA,DifParams$a,DifParams$b)

rU2<-rU1-rdelta

tiff("Fig3 Dif\_comparison.tiff", 500, 500)

plot(density(rU1), xlim=c(0.4, 0.7), ylim=c(0,20), main="", xlab="HRQoL", ylab="Density of simulated values")

lines(density(rU2), lty="dashed")

lines(density(rU2.raw), lwd=2, lty="dashed")

legend("topleft", legend=c("U1", "U2 using difference method", "U2 using independent sampling"), lwd=c(1,1,2), lty=c("solid", "dashed", "dashed"))

dev.off()

PSA.method10 <- data.frame(u1 = rU1, u2=rU2)

####################################### RESULTS ############################################

# want to plot scatter

# want bootstrapped estimates of means to compare

# packaging results together in list to make them easier to automate

MethodsBlock <- list(

methodboot=methodBoot.PSA,

method01=PSA.method01,

method02=PSA.method02,

method03=PSA.method03,

method04=PSA.method04,

method05=PSA.method05,

method06=PSA.method06,

method07=PSA.method07,

method08=PSA.method08,

method09=PSA.method09,

method10=PSA.method10)

# scatterplots

# do this as a single file

tiff("Fig4 PSA\_all.tiff", 900, 1200)

split.screen(c(4,3))

screen(1)

plot(u2 ~ u1, data=MethodsBlock$methodboot, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="a) Bootstrapped")

abline(0,1)

screen(3)

plot(u2 ~ u1, data=MethodsBlock$method01, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="b) Method 1")

abline(0,1)

screen(4)

plot(u2 ~ u1, data=MethodsBlock$method02, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="c) Method 2")

abline(0,1)

screen(5)

plot(u2 ~ u1, data=MethodsBlock$method03, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="d) Method 3")

abline(0,1)

screen(6)

plot(u2 ~ u1, data=MethodsBlock$method04, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="e) Method 4")

abline(0,1)

screen(7)

plot(u2 ~ u1, data=MethodsBlock$method05, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="f) Method 5")

abline(0,1)

screen(8)

plot(u2 ~ u1, data=MethodsBlock$method06, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="g) Method 6")

abline(0,1)

screen(9)

plot(u2 ~ u1, data=MethodsBlock$method07, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="h) Method 7")

abline(0,1)

screen(10)

plot(u2 ~ u1, data=MethodsBlock$method08, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="i) Method 8")

abline(0,1)

screen(11)

plot(u2 ~ u1, data=MethodsBlock$method09, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="j) Method 9")

abline(0,1)

screen(12)

plot(u2 ~ u1, data=MethodsBlock$method10, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="k) Method 10")

abline(0,1)

close.screen(1:12, T)

dev.off()

# # VIOLIN PLOTS

# Violin plots as one image

require(vioplot)

tiff("Fig5 Vioplot\_all.tiff", 1000, 1500)

split.screen(c(3,1))

screen(1)

# Distribution of U1s

vioplot(MethodsBlock[["methodboot"]]$u1,

MethodsBlock[["method01"]]$u1,

MethodsBlock[["method02"]]$u1,

MethodsBlock[["method03"]]$u1,

MethodsBlock[["method04"]]$u1,

MethodsBlock[["method05"]]$u1,

MethodsBlock[["method06"]]$u1,

MethodsBlock[["method07"]]$u1,

MethodsBlock[["method08"]]$u1,

MethodsBlock[["method09"]]$u1,

MethodsBlock[["method10"]]$u1,

# main="U1",

names=c("Boot", 1:10),

col="grey"

)

abline(v=1.5, lwd=2)

abline(h=mean(MethodsBlock[["methodboot"]]$u1), lty="dashed")

screen(2)

# Distribution of U2s

vioplot(MethodsBlock[["methodboot"]]$u2,

MethodsBlock[["method01"]]$u2,

MethodsBlock[["method02"]]$u2,

MethodsBlock[["method03"]]$u2,

MethodsBlock[["method04"]]$u2,

MethodsBlock[["method05"]]$u2,

MethodsBlock[["method06"]]$u2,

MethodsBlock[["method07"]]$u2,

MethodsBlock[["method08"]]$u2,

MethodsBlock[["method09"]]$u2,

MethodsBlock[["method10"]]$u2,

# main="U2",

names=c("Boot", 1:10),

col="grey"

)

abline(v=1.5, lwd=2)

abline(h=mean(MethodsBlock[["methodboot"]]$u2), lty="dashed")

screen(3)

# Distribution of differences

vioplot(with(MethodsBlock[["methodboot"]], (u1 - u2)),

with(MethodsBlock[["method01"]], (u1 - u2)),

with(MethodsBlock[["method02"]], (u1 - u2)),

with(MethodsBlock[["method03"]], (u1 - u2)),

with(MethodsBlock[["method04"]], (u1 - u2)),

with(MethodsBlock[["method05"]], (u1 - u2)),

with(MethodsBlock[["method06"]], (u1 - u2)),

with(MethodsBlock[["method07"]], (u1 - u2)),

with(MethodsBlock[["method08"]], (u1 - u2)),

with(MethodsBlock[["method09"]], (u1 - u2)),

with(MethodsBlock[["method10"]], (u1 - u2)),

# xlab="U1 - U2",

lwd=1.5,

names=c("Boot", 1:10),

col="grey"

)

abline(v=1.5, lwd=2)

abline(h=mean(with(MethodsBlock[["methodboot"]], (u1 - u2))), lty="dashed")

abline(h=0)

close.screen(all.screens=T)

dev.off()

############################################################################################

######### SUPPLEMENTARY ANALYSIS ###########################################################

############################################################################################

U1.mean <- 0.600

U2.mean <- 0.542

U1.sd <- 0.100

U2.sd <- 0.120

U1.N <- 80

U2.N <- 15

U1.se <- U1.sd / U1.N^0.5

U2.se <- U2.sd / U2.N^0.5

alt.U1.summary <- list(mu=U1.mean, sd=U1.se)

alt.U2.summary <- list(mu=U2.mean, sd=U2.se)

#plot(u2 ~ u1, data=PSA.method09)

# Method 8 & 9

tmp <- MakeBCVR.2d(alt.U1.summary$mu, alt.U1.summary$sd, alt.U2.summary$mu, alt.U2.summary$sd, n.psa=n.PSA, incBy=0.00001, upper=F)

tmp2 <- MakeBCVR.2d(alt.U1.summary$mu, alt.U1.summary$sd, alt.U2.summary$mu, alt.U2.summary$sd, n.psa=n.PSA, incBy=0.00001, upper=T)

PSA.method08.newData <-data.frame(tmp$samples)

names(PSA.method08.newData) <- c("u1", "u2")

method08.cov <- tmp$cov

method08.cor <- tmp$cor

method09.cov <- tmp2$cov

method09.cor <- tmp2$cor

PSA.method09.newData <- data.frame(tmp2$samples)

names(PSA.method09.newData) <- c("u1", "u2")

# METHOD 10: Beta distribution difference fitting

tmp <- getDifParam(alt.U1.summary$mu, alt.U1.summary$sd, alt.U2.summary$mu, alt.U2.summary$sd, F)

n.PSA <- 1000

alt.u1.param <- estBeta(alt.U1.summary$mu, alt.U1.summary$sd^2)

alt.u2.param <- estBeta(alt.U2.summary$mu, alt.U2.summary$sd^2)

#rU2.raw <- rnorm(n.PSA, U2.summary$mu, U2.summary$sd)

rU2.raw <- rbeta(n.PSA, u2.param$a, u2.param$b)

#rU1 <- rnorm(n.PSA, U1.summary$mu, U1.summary$sd)

rU1 <- rbeta(n.PSA, u1.param$a, u1.param$b)

rdelta<-rbeta(n.PSA,tmp$a,tmp$b)

rU2<-rU1-rdelta

# png("DensCompare.png", 500, 500)

# dev.off()

PSA.method10.newData <- data.frame(u1 = rU1, u2=rU2)

# plot(u2 ~ u1, data=PSA.method10.newData, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 10")

# abline(0,1)

tiff("Fig6 PSA\_alt.tiff", 800, 800)

split.screen(c(2,2))

screen(1)

plot(density(rU1), xlim=c(0.4, 0.7), ylim=c(0,20), main="a) Density plot", xlab="HRQoL", ylab="Density of simulated values")

lines(density(rU2), lty="dashed")

lines(density(rU2.raw), lwd=2, lty="dashed")

legend("topleft", legend=c("U1", "U2\*", "U2"), lwd=c(1,1,2), lty=c("solid", "dashed", "dashed"))

screen(2)

plot(u2 ~ u1, data=PSA.method08.newData, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="b) Method 8")

abline(0,1)

screen(3)

plot(u2 ~ u1, data=PSA.method09.newData, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="c) Method 9")

abline(0,1)

screen(4)

plot(u2 ~ u1, data=PSA.method10.newData, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="d) Method 10")

abline(0,1)

close.screen(1:4, T)

dev.off()

##############################################################################

##############################################################################

# Code below if producing the above graphs independently.

#png("PSA\_boot.png",300, 300)

#plot(u2 ~ u1, data=MethodsBlock$methodboot, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Bootstrapped")

#abline(0,1)

#dev.off()

png("PSA\_m01.png",300, 300)

plot(u2 ~ u1, data=MethodsBlock$method01, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), cex.main=0.6, main="Method 1")

abline(0,1)

dev.off()

png("PSA\_m02.png",300, 300)

plot(u2 ~ u1, data=MethodsBlock$method02, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 2")

abline(0,1)

dev.off()

png("PSA\_m03.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method03, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 3")

abline(0,1)

dev.off()

png("PSA\_m04.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method04, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 4")

abline(0,1)

dev.off()

png("PSA\_m05.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method05, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 5")

abline(0,1)

dev.off()

png("PSA\_m06.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method06, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 6")

abline(0,1)

dev.off()

png("PSA\_m07.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method07, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 7")

abline(0,1)

dev.off()

png("PSA\_m08.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method08, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 8")

abline(0,1)

dev.off()

png("PSA\_m09.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method09, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 9")

abline(0,1)

dev.off()

png("PSA\_m10.png", 300, 300)

plot(u2 ~ u1, data=MethodsBlock$method10, xlim=c(0.45, 0.7), ylim=c(0.45,0.7), main="Method 10")

abline(0,1)

dev.off()

######################################################################

#### IGNORE WHAT I HAVE BELOW:

# What I want:

# 1) Proportion of draws where monotonicity is violated

# 2) Mean value of U1

# 3) Mean value of U2

# 4) Mean of U1 - U2

# 3)

# VIOLATION OF MONOTONICITY

# 1) Proportion of draws where monotonicity is violated

PropViolations <- function(x){

return(length(which(x[,1] < x[,2])))

}

sapply(MethodsBlock, PropViolations)

# 2) Mean value of U1

# 3) Mean value of U2

tmp <- sapply(MethodsBlock, colMeans)

png("MeanScatter.png", 400, 400)

plot(NA, ylab="u2", xlab="u1", xlim=c(0.595, 0.605), ylim=c(0.545, 0.555))

points(tmp[2,1] ~ tmp[1,1], cex=3, pch=4, lwd=3)

for (i in 2:11){

points(tmp[2,i] ~ tmp[1,i], pch=i)

}

legend("topleft", legend=c("Bootstrap", 1:10), pch=c(4, 1:10))

dev.off()

tmp.difs <- rep(NA, 10)

for (i in 1:10){

tmp.difs[i] <- (

(tmp[1,i+1] - tmp[1,1])^2

+ (tmp[2,i+1] - tmp[2,1])^2

)^0.5

}

png("SdScatter.png", 400, 400)

tmp2 <- sapply(MethodsBlock, function(x) apply(x, 2, sd))

plot(NA, ylab="u2", xlab="u1", xlim=c(0.02, .024), ylim=c(00.02, .024))

points(tmp2[2,1] ~ tmp2[1,1], cex=3, pch=4, lwd=3)

for (i in 2:11){

points(tmp2[2,i] ~ tmp2[1,i], pch=i)

}

legend("bottomright", legend=c("Bootstrap", 1:10), pch=c(4, 1:10))

dev.off()

tmp2.difs <- rep(NA, 10)

for (i in 1:10){

tmp2.difs[i] <- (

(tmp2[1,i+1] - tmp2[1,1])^2

+ (tmp2[2,i+1] - tmp2[2,1])^2

)^0.5

}

tmp3 <- sapply(MethodsBlock, function(x) (cov(x)[2,1]))

tmp3.difs <- rep(NA, 10)

for (i in 1:10){tmp3.difs[i] <- abs(tmp3[i+1] - tmp3[1]) }

tmp4 <- sapply(MethodsBlock, function(x) (cor(x)[2,1]))

tmp4.difs <- rep(NA, 10)

for (i in 1:10){tmp4.difs[i] <- abs(tmp4[i+1] - tmp4[1]) }

# 4) Mean of U1 - U2

# now what is the distribution of differences u1 - u2?

difsDist <- matrix(NA, nrow=n.PSA, ncol=11)

for (i in 1:11){

difsDist[,i] <- MethodsBlock[[i]][,1] - MethodsBlock[[i]][,2]

}

names.quants.interest <- c("min", "lower", "median", "mean", "upper", "max")

quants.of.interest <- matrix(NA, nrow=length(names.quants.interest), ncol=11)

rownames(quants.of.interest) <- names.quants.interest

for (i in 1:11){

quants.of.interest["min", i] <- min(difsDist[,i])

quants.of.interest[c("lower", "median", "upper"),i] <- quantile(difsDist[,i], c(0.025, 0.5, 0.975))

quants.of.interest["max", i] <- max(difsDist[,i])

quants.of.interest["mean", i] <- mean(difsDist[,i])

}

colnames(quants.of.interest) <- c("boot", 1:10)