**An *R* Implementation of the Beta Density Weigh-Function Model**

Here, we present code for estimating the beta density weight-function model using *R*. The code should be placed into a series of text files defined below. The first file is a control file that requires one to input the data, set the starting values, and chose a fixed- or random-effects analysis. The file may be copy-pasted into *R* as a whole. Or, if the file is located on the desktop of the User, it may be invoked in *R* using the following source command:

source(“C:/User/Desktop/control.txt”) .

Follow the directions in the file closely. However, first make sure the function files (defined below) are in the appropriate folder and that *R* is set to the correct working directory. Once the control file is run, the following output will be printed:

1. Convergence diagnostics for the optimizer *nlminb* for both the unadjusted and adjusted models. (The output should be 0. If the output is 1, change the starting values and re-run the file.)
2. Unadjusted model: parameter estimates, standard errors, and the covariance matrix.
3. Adjusted model: parameter estimates, standard errors, and the covariance matrix.
4. Likelihood-ratio test results.
5. Heterogeneity assessment statistics: *Q*-within and *I2*.

**Mean-only Control File**

#In order to run the mean-only model analyses of the

#standard meta-analytic and beta density weight-function models,

#enter the data (i.e., effect sizes and variance), set the

#starting values, chose the appropriate inference model, and

#copy-paste this entire page of code into R.

#NOTE: Make sure the function files are in the appropriate folder

#and that the source path code is correct for and within each text file.

#An easy way to do this is to simply set the working directory to the

#folder that contains all the function files.

setwd("C:/User/Desktop/functions")

#Enter data. Define y as the effect size and v as the variance.

data <- read.table("C:/User/Desktop/data.txt", col.names=c("id","es","v"))

y <- data$es

v <- data$v

#Set starting values.

a <- 1

b <- 1

vc <- 0 #Set to zero if estimating a fixed-effects model.

beta0 <- .5

#Chose whether to run a fixed- or random-effects analysis:

#Fixed-effects. (Note: must set vc to zero above.)

#source("FE.models.txt")

#Random-effects.

source("RE.models.txt")

#Do not modify below this line.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Estimate the p-values.

#NOTE: This includes the p-value adjustment that allows the adjusted model to run.

p <- 1-pnorm(y/sqrt(v))

for(l in 1:length(p)) {

p[l] <- if (p[l] < .00001) .00001

else if (p[l] > .99999) .99999

else p[l] }

#Fit the unadjusted standard meta-analytic model.

unadjustedREpars <- c(vc,beta0)

REunadjusted <- nlminb(start=unadjustedREpars,objective=neglikeunadjustedRE,lower=c(0,-Inf),control=list(eval.max=1000,iter.max=1000,abs.tol=10e-5,rel.tol=10e-5))

REunadjustedest <- REunadjusted$par

#Fit the adjusted beta density weight-function model.

adjustedREpars <- c(a,b,vc,beta0)

REadjusted <- nlminb(start=adjustedREpars,objective=neglikeadjustedRE,lower=c(0,0,0,-Inf),control=list(eval.max=1000,iter.max=1000,abs.tol=10e-5,rel.tol=10e-5))

REadjustedest <- REadjusted$par

#Estimate the standard errors of the parameter estimates for both models.

source("SERE.calculation.txt")

#Estimate the likelihood-ratio test for publication bias.

LRRE <- -2\*(REadjusted$obj - REunadjusted$obj)

dfLRRE <- length(adjustedREpars) - length(unadjustedREpars)

pLRRE <- 1-pchisq(LRRE,dfLRRE)

#Estimate the data variability using Q-within and I-squared.

source("variability.calculation.txt")

print("Assess convergence:"); print("Unadjusted model Adjusted model"); print(c(REunadjusted$conv,REadjusted$conv))

print("Unadjusted parameter estimates:"); print("vc beta0"); print(REunadjustedest)

print("Standard errors:"); print(UnadjustedRESEs)

print("Covariance matrix:"); print(UnadjustedREcovmat)

print("Adjusted parameter estimates:"); print("a b vc beta0"); print(REadjustedest)

print("Standard errors:"); print(AdjustedRESEs)

print("Covariance matrix:"); print(AdjustedREcovmat)

print("Likelihood-ratio test:"); print("Chi-squared estimate, degrees of freedom, and p-value"); print(c(LRRE,dfLRRE,pLRRE))

print("Q-within estimate, degrees of freedom, and p-value:"); print(c(Qw,dfQw,pQw))

print("I-squared estimate:"); print(I2)

**Moderator Control File**

#In order to run the moderator analyses of the

#standard meta-analytic and beta density weight-function models,

#enter the data (i.e., effect sizes, variance, and moderator), set the

#starting values, chose the appropriate inference model, and

#copy-paste this entire page of code into R.

#NOTE: Make sure the function files are in the appropriate folder

#and that the source path code is correct for and within each text file.

#An easy way to do this is to simply set the working directory to the

#folder that contains all the function files.

setwd("C:/User/Desktop/functions")

#Enter data. Define y as the effect size, v as the variance, and x as the moderator.

data <- read.table("C:/User/Desktop/data.txt", col.names=c("id","es","v","mod"))

y <- data$es

v <- data$v

x <- data$mod

#Set starting values.

a <- 1

b <- 1

vc <- 0 #Set to zero if estimating a fixed-effects model.

beta0 <- .5

beta1 <- .5

#Chose whether to run a fixed- or mixed-effects analysis:

#Fixed-effects. (Note: must set vc to zero above.)

#source("FEC.models.txt")

#Random-effects.

source("ME.models.txt")

#Do not modify below this line.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#Estimate the p-values.

#NOTE: This includes the p-value adjustment that allows the adjusted model to run.

p <- 1-pnorm(y/sqrt(v))

for(l in 1:length(p)) {

p[l] <- if (p[l] < .00001) .00001

else if (p[l] > .99999) .99999

else p[l] }

#Set the predictor matrix.

X <- cbind(rep(1,length(y)),x)

#Fit the unadjusted standard meta-analytic model.

unadjustedMEpars <- c(vc,beta0,beta1)

MEunadjusted <- nlminb(start=unadjustedMEpars,objective=neglikeunadjustedME,lower=c(0,-Inf,-Inf),control=list(eval.max=1000,iter.max=1000,abs.tol=10e-5,rel.tol=10e-5))

MEunadjustedest <- MEunadjusted$par

#Fit the adjusted beta density weight-function model.

adjustedMEpars <- c(a,b,vc,beta0,beta1)

MEadjusted <- nlminb(start=adjustedMEpars,objective=neglikeadjustedME,lower=c(0,0,0,-Inf,-Inf),control=list(eval.max=1000,iter.max=1000,abs.tol=10e-5,rel.tol=10e-5))

MEadjustedest <- MEadjusted$par

#Estimate the standard errors of the parameter estimates for both models.

source("SEME.calculation.txt")

#Estimate the likelihood-ratio test for publication bias.

LRME <- -2\*(MEadjusted$obj - MEunadjusted$obj)

dfLRME <- length(adjustedMEpars) - length(unadjustedMEpars)

pLRME <- 1-pchisq(LRME,dfLRME)

#Estimate the data variability using Q-within and I-squared.

source("variability.calculation.txt")

print("Assess convergence:"); print("Unadjusted model Adjusted model"); print(c(MEunadjusted$conv,MEadjusted$conv))

print("Unadjusted parameter estimates:"); print("vc beta0 beta1"); print(MEunadjustedest)

print("Standard errors:"); print(UnadjustedMESEs)

print("Covariance matrix:"); print(UnadjustedMEcovmat)

print("Adjusted parameter estimates:"); print("a b vc beta0 beta1"); print(MEadjustedest)

print("Standard errors:"); print(AdjustedMESEs)

print("Covariance matrix:"); print(AdjustedMEcovmat)

print("Likelihood-ratio test:"); print("Chi-squared estimate, degrees of freedom, and p-value"); print(c(LRME,dfLRME,pLRME))

print("Q-within estimate, degrees of freedom, and p-value:"); print(c(Qw,dfQw,pQw))

print("I-squared estimate:"); print(I2)

**Functions Files**

The function files should be located in a folder named “functions” on the User’s desktop. These files should NOT be altered in any way.

The following text should appear in a file named “FE.models.txt.”

#

#Negative likelihood functions of the fixed-effects models.

#Unadjusted mean-only model.

neglikeunadjustedRE <- function(unadjustedREpars) {

vc2 <- 0

beta02 <- unadjustedREpars[2]

thisone <- 1/2\*log(v+vc2) + 1/2\*((y-beta02)^2/(v+vc2))

return(sum(thisone)) }

#Adjusted mean-only beta density weight-function model.

denfxn <- function(yy,vv,a2,b2,vc2,beta02) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,beta02,sqrt(vv+vc2))

return(part1\*part2\*part3) }

neglikeadjustedRE <- function(adjustedREpars) {

a2 <- adjustedREpars[1]

b2 <- adjustedREpars[2]

vc2 <- 0

beta02 <- adjustedREpars[4]

num <- sum( (a2-1)\*log(p) +

(b2-1)\*log(1-p) -

1/2\*log(v+vc2) - 1/2\*((y-beta02)^2/(v+vc2)) )

total <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

answer <- integrate(denfxn,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,beta02=beta02)

bottom <- .00001\*denfxn(-Inf,vv,a2,b2,vc2,beta02)

top <- .00001\*denfxn(Inf,vv,a2,b2,vc2,beta02)

total[i] <- log( answer$value + bottom + top ) }

den <- sum(total)

return(-num+den) }

#

The following text should appear in a file named “FEC.models.txt.”

#

#Negative likelihood functions of the moderator fixed-effects models.

#Unadjusted moderator model.

neglikeunadjustedME <- function(unadjustedMEpars) {

vc2 <- 0

beta02 <- unadjustedMEpars[2]

beta12 <- unadjustedMEpars[3]

thisone <- 1/2\*log(v+vc2) + 1/2 \*((y-X%\*%c(beta02,beta12))^2/(v+vc2))

return(sum(thisone)) }

#Adjusted moderator beta density weight-function model.

denfxn <- function(yy,vv,XX,a2,b2,vc2,beta02,beta12) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(beta02,beta12),sqrt(vv+vc2))

return(part1\*part2\*part3) }

neglikeadjustedME <- function(adjustedMEpars) {

a2 <- adjustedMEpars[1]

b2 <- adjustedMEpars[2]

vc2 <- 0

beta02 <- adjustedMEpars[4]

beta12 <- adjustedMEpars[5]

num <- sum( (a2-1)\*log(p) +

(b2-1)\*log(1-p) -

1/2\*log(v+vc2) - 1/2\*((y-X%\*%c(beta02,beta12))^2/(v+vc2)) )

total <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

answer <- integrate(denfxn,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,beta02=beta02,beta12=beta12)

bottom <- .00001\*denfxn(-Inf,vv,XX,a2,b2,vc2,beta02,beta12)

top <- .00001\*denfxn(Inf,vv,XX,a2,b2,vc2,beta02,beta12)

total[i] <- log( answer$value + bottom + top ) }

den <- sum(total)

return(-num+den) }

#

The following text should appear in a file named “ME.models.txt.”

#

#Negative likelihood functions of the mixed-effects models.

#Unadjusted moderator model.

neglikeunadjustedME <- function(unadjustedMEpars) {

vc2 <- unadjustedMEpars[1]

beta02 <- unadjustedMEpars[2]

beta12 <- unadjustedMEpars[3]

thisone <- 1/2\*log(v+vc2) + 1/2 \*((y-X%\*%c(beta02,beta12))^2/(v+vc2))

return(sum(thisone)) }

#Adjusted moderator beta density weight-function model.

denfxn <- function(yy,vv,XX,a2,b2,vc2,beta02,beta12) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(beta02,beta12),sqrt(vv+vc2))

return(part1\*part2\*part3) }

neglikeadjustedME <- function(adjustedMEpars) {

a2 <- adjustedMEpars[1]

b2 <- adjustedMEpars[2]

vc2 <- adjustedMEpars[3]

beta02 <- adjustedMEpars[4]

beta12 <- adjustedMEpars[5]

num <- sum( (a2-1)\*log(p) +

(b2-1)\*log(1-p) -

1/2\*log(v+vc2) - 1/2\*((y-X%\*%c(beta02,beta12))^2/(v+vc2)) )

total <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

answer <- integrate(denfxn,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,beta02=beta02,beta12=beta12)

bottom <- .00001\*denfxn(-Inf,vv,XX,a2,b2,vc2,beta02,beta12)

top <- .00001\*denfxn(Inf,vv,XX,a2,b2,vc2,beta02,beta12)

total[i] <- log( answer$value + bottom + top ) }

den <- sum(total)

return(-num+den) }

#

The following text should appear in a file named “RE.models.txt.”

#

#Negative likelihood functions of the random-effects models.

#Unadjusted mean-only model.

neglikeunadjustedRE <- function(unadjustedREpars) {

vc2 <- unadjustedREpars[1]

beta02 <- unadjustedREpars[2]

thisone <- 1/2\*log(v+vc2) + 1/2\*((y-beta02)^2/(v+vc2))

return(sum(thisone)) }

#Adjusted mean-only beta density weight-function model.

denfxn <- function(yy,vv,a2,b2,vc2,beta02) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,beta02,sqrt(vv+vc2))

return(part1\*part2\*part3) }

neglikeadjustedRE <- function(adjustedREpars) {

a2 <- adjustedREpars[1]

b2 <- adjustedREpars[2]

vc2 <- adjustedREpars[3]

beta02 <- adjustedREpars[4]

num <- sum( (a2-1)\*log(p) +

(b2-1)\*log(1-p) -

1/2\*log(v+vc2) - 1/2\*((y-beta02)^2/(v+vc2)) )

total <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

answer <- integrate(denfxn,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,beta02=beta02)

bottom <- .00001\*denfxn(-Inf,vv,a2,b2,vc2,beta02)

top <- .00001\*denfxn(Inf,vv,a2,b2,vc2,beta02)

total[i] <- log( answer$value + bottom + top ) }

den <- sum(total)

return(-num+den) }

#

The following text should appear in a file named “SEFE.adjusted.txt.”

#

#1) Input functions.

source("SERE.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEFEadjustedest) {

a2 <- SEFEadjustedest[1]

b2 <- SEFEadjustedest[2]

vc2 <- 0

mu2 <- SEFEadjustedest[3]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dlda <- sum( log(p) - da/A )

dldb <- sum( log(1-p) - db/A )

# dldvc <- sum( ((y-mu2)^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2)/(v+vc2) - dmu/A )

return( matrix(c(dlda,dldb,dldmu),1,3) ) }

#3) Calculate the hessian.

hessian <- function(SEFEadjustedest) {

a2 <- SEFEadjustedest[1]

b2 <- SEFEadjustedest[2]

vc2 <- 0

mu2 <- SEFEadjustedest[3]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

da2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

db2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dadb <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dadvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dadmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dbdvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dbdmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvcdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

d2lda2 <- sum ( da^2/A^2 - da2/A )

d2ldadb <- sum ( da\*db/A^2 - dadb/A )

# d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

d2ldb2 <- sum ( db^2/A^2 - db2/A )

# d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

# d2ldvc2 <- sum ( (-1\*((y-mu2)^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

# d2ldvcdmu <- sum ( (-1\*(y-mu2)/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

return( matrix(c(d2lda2,d2ldadb,d2ldadmu,

d2ldadb,d2ldb2,d2ldbdmu,

d2ldadmu,d2ldbdmu,d2ldmu2),3,3) ) }

hess <- hessian(SEFEadjustedest)

covmat <- solve(-hess)

SEa <- sqrt(covmat[1,1])

SEb <- sqrt(covmat[2,2])

SEvc <- -99

SEb0 <- sqrt(covmat[3,3])

FEadjustedSEs <- c(SEa,SEb,SEvc,SEb0)

FEadjustedcovmat <- covmat

#

The following text should appear in a file named “SEFE.unadjusted.”

#

#1) Input functions.

source("SERE.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEFEunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- 0

mu2 <- SEFEunadjustedest[1]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dlda <- sum( log(p) - da/A )

# dldb <- sum( log(1-p) - db/A )

# dldvc <- sum( ((y-mu2)^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2)/(v+vc2) - dmu/A )

return( matrix(c(dldmu),1,1) ) }

#3) Calculate the hessian.

hessian <- function(SEFEunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- 0

mu2 <- SEFEunadjustedest[1]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# da2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# db2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvc2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dadb <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dadvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dadmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dbdvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dbdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# dvcdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,a2=a2,b2=b2,vc2=vc2,mu2=mu2)$value }

# d2lda2 <- sum ( da^2/A^2 - da2/A )

# d2ldadb <- sum ( da\*db/A^2 - dadb/A )

# d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

# d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

# d2ldb2 <- sum ( db^2/A^2 - db2/A )

# d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

# d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

# d2ldvc2 <- sum ( (-1\*((y-mu2)^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

# d2ldvcdmu <- sum ( (-1\*(y-mu2)/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

return( matrix(c(d2ldmu2),1,1) ) }

hess <- hessian(SEFEunadjustedest)

covmat <- solve(-hess)

SEa <- -99

SEb <- -99

SEvc <- -99

SEb0 <- sqrt(covmat[1,1])

FEunadjustedSEs <- c(SEvc,SEb0)

FEunadjustedcovmat <- covmat

#

The following text should appear in a file named “SEFEC.adjusted.txt.”

#

#1) Input functions.

source("SEME.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEFECadjustedest) {

a2 <- SEFECadjustedest[1]

b2 <- SEFECadjustedest[2]

vc2 <- 0

mu2 <- SEFECadjustedest[3]

slope2 <- SEFECadjustedest[4]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dlda <- sum( log(p) - da/A )

dldb <- sum( log(1-p) - db/A )

# dldvc <- sum( ((y-mu2-slope2\*X[,2])^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2-slope2\*X[,2])/(v+vc2) - dmu/A )

dldslope <- sum( X[,2]\*(y-mu2-slope2\*X[,2])/(v+vc2) - dslope/A )

return( matrix(c(dlda,dldb,dldmu,dldslope),1,4) ) }

#3) Calculate the hessian.

hessian <- function(SEFECadjustedest) {

a2 <- SEFECadjustedest[1]

b2 <- SEFECadjustedest[2]

vc2 <- 0

mu2 <- SEFECadjustedest[3]

slope2 <- SEFECadjustedest[4]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope2[i] <- integrate(fxndslope2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadb <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadslope[i] <- integrate(fxndadslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dbdmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dbdslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dbdslope[i]<- integrate(fxndbdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvcdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvcdslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvcdslope[i] <- integrate(fxndvcdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmudslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmudslope[i] <- integrate(fxndmudslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

d2lda2 <- sum ( da^2/A^2 - da2/A )

d2ldadb <- sum ( da\*db/A^2 - dadb/A )

# d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

d2ldadslope <- sum ( da\*dslope/A^2 - dadslope/A )

d2ldb2 <- sum ( db^2/A^2 - db2/A )

# d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

d2ldbdslope <- sum ( db\*dslope/A^2 - dbdslope/A )

# d2ldvc2 <- sum ( (-1\*((y-mu2-slope2\*X[,2])^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

# d2ldvcdmu <- sum ( (-1\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

# d2ldvcdslope <- sum ( (-X[,2]\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dslope/A^2 - dvcdslope/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

d2ldmudslope <- sum ( -X[,2]/(v+vc2) + dmu\*dslope/A^2 - dmudslope/A )

d2ldslope2 <- sum ( -(X[,2]^2)/(v+vc2) + dslope^2/A^2 - dslope2/A )

return( matrix(c(d2lda2,d2ldadb,d2ldadmu,d2ldadslope,

d2ldadb,d2ldb2,d2ldbdmu,d2ldbdslope,

d2ldadmu,d2ldbdmu,d2ldmu2,d2ldmudslope,

d2ldadslope,d2ldbdslope,d2ldmudslope,d2ldslope2),4,4) ) }

hess <- hessian(SEFECadjustedest)

covmat <- solve(-hess)

SEa <- sqrt(covmat[1,1])

SEb <- sqrt(covmat[2,2])

SEvc <- -99

SEb0 <- sqrt(covmat[3,3])

SEb1 <- sqrt(covmat[4,4])

FECadjustedSEs <- c(SEa,SEb,SEvc,SEb0,SEb1)

FECadjustedcovmat <- covmat

#

The following text should appear in a file named “SEFEC.unadjusted.txt.”

#

#1) Input functions.

source("SEME.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEFECunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- 0

mu2 <- SEFECunadjustedest[1]

slope2 <- SEFECunadjustedest[2]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dlda <- sum( log(p) - da/A )

# dldb <- sum( log(1-p) - db/A )

# dldvc <- sum( ((y-mu2-slope2\*X[,2])^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2-slope2\*X[,2])/(v+vc2) - dmu/A )

dldslope <- sum( X[,2]\*(y-mu2-slope2\*X[,2])/(v+vc2) - dslope/A )

return( matrix(c(dldmu,dldslope),1,2) ) }

#3) Calculate the hessian.

hessian <- function(SEFECunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- 0

mu2 <- SEFECunadjustedest[1]

slope2 <- SEFECunadjustedest[2]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvc2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope2[i] <- integrate(fxndslope2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadb <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadslope[i] <- integrate(fxndadslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdslope[i]<- integrate(fxndbdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvcdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dvcdslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dvcdslope[i] <- integrate(fxndvcdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmudslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmudslope[i] <- integrate(fxndmudslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# d2lda2 <- sum ( da^2/A^2 - da2/A )

# d2ldadb <- sum ( da\*db/A^2 - dadb/A )

# d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

# d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

# d2ldadslope <- sum ( da\*dslope/A^2 - dadslope/A )

# d2ldb2 <- sum ( db^2/A^2 - db2/A )

# d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

# d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

# d2ldbdslope <- sum ( db\*dslope/A^2 - dbdslope/A )

# d2ldvc2 <- sum ( (-1\*((y-mu2-slope2\*X[,2])^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

# d2ldvcdmu <- sum ( (-1\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

# d2ldvcdslope <- sum ( (-X[,2]\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dslope/A^2 - dvcdslope/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

d2ldmudslope <- sum ( -X[,2]/(v+vc2) + dmu\*dslope/A^2 - dmudslope/A )

d2ldslope2 <- sum ( -(X[,2]^2)/(v+vc2) + dslope^2/A^2 - dslope2/A )

return( matrix(c(d2ldmu2,d2ldmudslope,

d2ldmudslope,d2ldslope2),2,2) ) }

hess <- hessian(SEFECunadjustedest)

covmat <- solve(-hess)

SEa <- -99

SEb <- -99

SEvc <- -99

SEb0 <- sqrt(covmat[1,1])

SEb1 <- sqrt(covmat[2,2])

FECunadjustedSEs <- c(SEvc,SEb0,SEb1)

FECunadjustedcovmat <- covmat

#

The following text should appear in a file named “SEME.adjusted.txt.”

#

#1) Input functions.

source("SEME.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEMEadjustedest) {

a2 <- SEMEadjustedest[1]

b2 <- SEMEadjustedest[2]

vc2 <- SEMEadjustedest[3]

mu2 <- SEMEadjustedest[4]

slope2 <- SEMEadjustedest[5]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dlda <- sum( log(p) - da/A )

dldb <- sum( log(1-p) - db/A )

dldvc <- sum( ((y-mu2-slope2\*X[,2])^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2-slope2\*X[,2])/(v+vc2) - dmu/A )

dldslope <- sum( X[,2]\*(y-mu2-slope2\*X[,2])/(v+vc2) - dslope/A )

return( matrix(c(dlda,dldb,dldvc,dldmu,dldslope),1,5) ) }

#3) Calculate the hessian.

hessian <- function(SEMEadjustedest) {

a2 <- SEMEadjustedest[1]

b2 <- SEMEadjustedest[2]

vc2 <- SEMEadjustedest[3]

mu2 <- SEMEadjustedest[4]

slope2 <- SEMEadjustedest[5]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

da2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

db2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope2[i] <- integrate(fxndslope2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadb <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dadslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dadslope[i] <- integrate(fxndadslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dbdvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dbdmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dbdslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dbdslope[i]<- integrate(fxndbdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvcdmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvcdslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvcdslope[i] <- integrate(fxndvcdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmudslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmudslope[i] <- integrate(fxndmudslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

d2lda2 <- sum ( da^2/A^2 - da2/A )

d2ldadb <- sum ( da\*db/A^2 - dadb/A )

d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

d2ldadslope <- sum ( da\*dslope/A^2 - dadslope/A )

d2ldb2 <- sum ( db^2/A^2 - db2/A )

d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

d2ldbdslope <- sum ( db\*dslope/A^2 - dbdslope/A )

d2ldvc2 <- sum ( (-1\*((y-mu2-slope2\*X[,2])^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

d2ldvcdmu <- sum ( (-1\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

d2ldvcdslope <- sum ( (-X[,2]\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dslope/A^2 - dvcdslope/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

d2ldmudslope <- sum ( -X[,2]/(v+vc2) + dmu\*dslope/A^2 - dmudslope/A )

d2ldslope2 <- sum ( -(X[,2]^2)/(v+vc2) + dslope^2/A^2 - dslope2/A )

return( matrix(c(d2lda2,d2ldadb,d2ldadvc,d2ldadmu,d2ldadslope,

d2ldadb,d2ldb2,d2ldbdvc,d2ldbdmu,d2ldbdslope,

d2ldadvc,d2ldbdvc,d2ldvc2,d2ldvcdmu,d2ldvcdslope,

d2ldadmu,d2ldbdmu,d2ldvcdmu,d2ldmu2,d2ldmudslope,

d2ldadslope,d2ldbdslope,d2ldvcdslope,d2ldmudslope,d2ldslope2),5,5) ) }

hess <- hessian(SEMEadjustedest)

covmat <- solve(-hess)

SEa <- sqrt(covmat[1,1])

SEb <- sqrt(covmat[2,2])

SEvc <- sqrt(covmat[3,3])

SEb0 <- sqrt(covmat[4,4])

SEb1 <- sqrt(covmat[5,5])

MEadjustedSEs <- c(SEa,SEb,SEvc,SEb0,SEb1)

MEadjustedcovmat <- covmat

#

The following text should appear in a file named “SEME.calculation.txt.”

#

#Unadjusted ME model.

SEFECunadjustedest <- c(MEunadjustedest[2],MEunadjustedest[3])

SEMEunadjustedest <- c(MEunadjustedest[1],MEunadjustedest[2],MEunadjustedest[3])

if(MEunadjustedest[1]==0) source("SEFEC.unadjusted.txt") else source("SEME.unadjusted.txt")

if(MEunadjustedest[1]==0) UnadjustedMESEs <- FECunadjustedSEs else UnadjustedMESEs <- MEunadjustedSEs

if(MEunadjustedest[1]==0) UnadjustedMEcovmat <- FECunadjustedcovmat else UnadjustedMEcovmat <- MEunadjustedcovmat

#Adjusted ME model.

SEFECadjustedest <- c(MEadjustedest[1],MEadjustedest[2],MEadjustedest[4],MEadjustedest[5])

SEMEadjustedest <- c(MEadjustedest[1],MEadjustedest[2],MEadjustedest[3],MEadjustedest[4],MEadjustedest[5])

if(MEadjustedest[3]==0) source("SEFEC.adjusted.txt") else source("SEME.adjusted.txt")

if(MEadjustedest[3]==0) AdjustedMESEs <- FECadjustedSEs else AdjustedMESEs <- MEadjustedSEs

if(MEadjustedest[3]==0) AdjustedMEcovmat <- FECadjustedcovmat else AdjustedMEcovmat <- MEadjustedcovmat

#

The following text should appear in a file named “SEME.functions.txt.”

#

#Note: fxnA is the same as denfxn.

fxnA <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

return(part1\*part2\*part3) }

fxnda <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)

return(part1\*part2\*part3\*part4) }

fxndb <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(1-pp)

return(part1\*part2\*part3\*part4) }

fxndvc <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- ((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2)

return(part1\*part2\*part3\*part4) }

fxndmu <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- (yy-mu2-slope2\*XX[2])/(vv+vc2)

return(part1\*part2\*part3\*part4) }

fxndslope <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2)

return(part1\*part2\*part3\*part4) }

fxnda2 <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)^2

return(part1\*part2\*part3\*part4) }

fxndb2 <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(1-pp)^2

return(part1\*part2\*part3\*part4) }

fxndvc2 <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- ( ((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2) )^2

part5 <- ((yy-mu2-slope2\*XX[2])^2)/((vv+vc2)^3)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

fxndmu2 <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- ( (yy-mu2-slope2\*XX[2])/(vv+vc2) )^2

part5 <- 1/(vv+vc2)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

fxndslope2 <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- ( XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2) )^2

part5 <- (XX[2]^2)/(vv+vc2)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

fxndadb <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)\*log(1-pp)

return(part1\*part2\*part3\*part4) }

fxndadvc <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)\*(((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2))

return(part1\*part2\*part3\*part4) }

fxndadmu <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)\*((yy-mu2-slope2\*XX[2])/(vv+vc2))

return(part1\*part2\*part3\*part4) }

fxndadslope <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(pp)\*(XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2))

return(part1\*part2\*part3\*part4) }

fxndbdvc <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(1-pp)\*(((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2))

return(part1\*part2\*part3\*part4) }

fxndbdmu <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(1-pp)\*((yy-mu2-slope2\*XX[2])/(vv+vc2))

return(part1\*part2\*part3\*part4) }

fxndbdslope <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- log(1-pp)\*(XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2))

return(part1\*part2\*part3\*part4) }

fxndvcdmu <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- (((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2))\*((yy-mu2-slope2\*XX[2])/(vv+vc2))

part5 <- (yy-mu2-slope2\*XX[2])/((vv+vc2)^2)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

fxndvcdslope <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- (((yy-mu2-slope2\*XX[2])^2)/(2\*(vv+vc2)^2))\*(XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2))

part5 <- XX[2]\*(yy-mu2-slope2\*XX[2])/((vv+vc2)^2)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

fxndmudslope <- function(yy,vv,XX,a2,b2,vc2,mu2,slope2) {

pp <- 1-pnorm(yy/sqrt(vv))

pp <- ifelse(pp < .00001,.00001,pp)

pp <- ifelse(pp > .99999,.99999,pp)

part1 <- pp^(a2-1)

part2 <- (1-pp)^(b2-1)

part3 <- dnorm(yy,XX%\*%c(mu2,slope2),sqrt(vv+vc2))

part4 <- (XX[2]\*(yy-mu2-slope2\*XX[2])/(vv+vc2))\*((yy-mu2-slope2\*XX[2])/(vv+vc2))

part5 <- XX[2]/(vv+vc2)

return(part1\*part2\*part3\*part4 - part1\*part2\*part3\*part5) }

#

The following text should appear in a file named “SEME.unadjusted.txt.”

#

#1) Input functions.

source("SEME.functions.txt")

#2) Calculate the gradient.

gradient <- function(SEMEunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- SEMEunadjustedest[1]

mu2 <- SEMEunadjustedest[2]

slope2 <- SEMEunadjustedest[3]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dlda <- sum( log(p) - da/A )

# dldb <- sum( log(1-p) - db/A )

dldvc <- sum( ((y-mu2-slope2\*X[,2])^2)/(2\*(v+vc2)^2) - dvc/A )

dldmu <- sum( (y-mu2-slope2\*X[,2])/(v+vc2) - dmu/A )

dldslope <- sum( X[,2]\*(y-mu2-slope2\*X[,2])/(v+vc2) - dslope/A )

return( matrix(c(dldvc,dldmu,dldslope),1,3) ) }

#3) Calculate the hessian.

hessian <- function(SEMEunadjustedest) {

a2 <- 1

b2 <- 1

vc2 <- SEMEunadjustedest[1]

mu2 <- SEMEunadjustedest[2]

slope2 <- SEMEunadjustedest[3]

A <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

A[i] <- integrate(fxnA,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da[i] <- integrate(fxnda,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db[i]<- integrate(fxndb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc[i] <- integrate(fxndvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu[i] <- integrate(fxndmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope[i] <- integrate(fxndslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# da2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# da2[i] <- integrate(fxnda2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# db2 <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# db2[i]<- integrate(fxndb2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvc2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvc2[i] <- integrate(fxndvc2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmu2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmu2[i] <- integrate(fxndmu2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dslope2 <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dslope2[i] <- integrate(fxndslope2,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadb <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadb[i] <- integrate(fxndadb,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadvc[i] <- integrate(fxndadvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadmu[i] <- integrate(fxndadmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dadslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dadslope[i] <- integrate(fxndadslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdvc <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdvc[i]<- integrate(fxndbdvc,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdmu <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdmu[i]<- integrate(fxndbdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# dbdslope <- rep(0,length(y))

# for(i in 1:length(y)) {

# vv = v[i]

# yy = y[i]

# XX = X[i,]

# dbdslope[i]<- integrate(fxndbdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvcdmu <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvcdmu[i] <- integrate(fxndvcdmu,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dvcdslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dvcdslope[i] <- integrate(fxndvcdslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

dmudslope <- rep(0,length(y))

for(i in 1:length(y)) {

vv = v[i]

yy = y[i]

XX = X[i,]

dmudslope[i] <- integrate(fxndmudslope,-Inf,Inf,vv=vv,XX,a2=a2,b2=b2,vc2=vc2,mu2=mu2,slope2=slope2)$value }

# d2lda2 <- sum ( da^2/A^2 - da2/A )

# d2ldadb <- sum ( da\*db/A^2 - dadb/A )

# d2ldadvc <- sum ( da\*dvc/A^2 - dadvc/A )

# d2ldadmu <- sum ( da\*dmu/A^2 - dadmu/A )

# d2ldadslope <- sum ( da\*dslope/A^2 - dadslope/A )

# d2ldb2 <- sum ( db^2/A^2 - db2/A )

# d2ldbdvc <- sum ( db\*dvc/A^2 - dbdvc/A )

# d2ldbdmu <- sum ( db\*dmu/A^2 - dbdmu/A )

# d2ldbdslope <- sum ( db\*dslope/A^2 - dbdslope/A )

d2ldvc2 <- sum ( (-1\*((y-mu2-slope2\*X[,2])^2)/((v+vc2)^3)) + dvc^2/A^2 - dvc2/A )

d2ldvcdmu <- sum ( (-1\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dmu/A^2 - dvcdmu/A )

d2ldvcdslope <- sum ( (-X[,2]\*(y-mu2-slope2\*X[,2])/((v+vc2)^2)) + dvc\*dslope/A^2 - dvcdslope/A )

d2ldmu2 <- sum ( -1/(v+vc2) + dmu^2/A^2 - dmu2/A )

d2ldmudslope <- sum ( -X[,2]/(v+vc2) + dmu\*dslope/A^2 - dmudslope/A )

d2ldslope2 <- sum ( -(X[,2]^2)/(v+vc2) + dslope^2/A^2 - dslope2/A )

return( matrix(c(d2ldvc2,d2ldvcdmu,d2ldvcdslope,

d2ldvcdmu,d2ldmu2,d2ldmudslope,

d2ldvcdslope,d2ldmudslope,d2ldslope2),3,3) ) }

hess <- hessian(SEMEunadjustedest)

covmat <- solve(-hess)

SEa <- -99

SEb <- -99

SEvc <- sqrt(covmat[1,1])

SEb0 <- sqrt(covmat[2,2])

SEb1 <- sqrt(covmat[3,3])

MEunadjustedSEs <- c(SEvc,SEb0,SEb1)

MEunadjustedcovmat <- covmat

#

The following text should appear in a file named “SERE.adjusted.txt.”

#

#

The following text should appear in a file named “SERE.calculation.txt.”

#

#

The following text should appear in a file named “SERE.functions.txt.”

#

#

The following text should appear in a file named “SERE.unadjusted.txt.”

#

#

The following text should appear in a file named “variability.calculation.txt.”

#

#