#This program estimates the GEMM for all ages and non-accidental deaths

#load necessary R packages

library(minpack.lm)

library(Matrix)

library(metafor)

#define threshold concentration THRES, maximum concentration for plotting M, plotting increment bb

#read in 2 files for each of the 15 cohorts with subject level data

#results file gives model parameter estimates and weights

# ap\_exp file gives concentrations for RR prediction

setwd ("C:/GEMM files")

dataACS<- read.csv("ACS results.csv", header = T)

dataxACS= read.csv("ACS ap\_exp.csv", header = T)

dataRome<- read.csv("Rome results.csv ", header = T)

dataxRome= read.csv("Rome ap.exp.csv ", header = T)

dataChinaH<- read.csv("China results.csv ", header=T)

dataChinaL<- read.csv("China IND COVARIATES results.csv ", header=T)

dataxChina= read.csv("China ap\_exp.csv", header = T)

dataHK<- read.csv("Hong Konk results.csv", header = T)

dataxHK= read.csv("Hong Kong ap\_exp.csv", header = T)

dataUK<- read.csv("England results.csv", header = T)

dataxUK= read.csv("England ap\_exp.csv", header = T)

dataCCHS<- read.csv("CCHS results.csv", header = T)

dataxCCHS<- read.csv("CCHS ap\_exp.csv", header = T)

dataAARP<- read.csv("AARP results.csv", header = T)

dataxAARP= read.csv("AARP ap\_exp.csv", header = T)

dataCanCHEC2001<- read.csv("CanCHEC 2001 results.csv", header = T)

dataxCanCHEC2001= read.csv("CanCHEC 2001 ap\_exp.csv", header = T)

dataBreast<- read.csv("Breast results.csv", header = T)

dataxBreast<- read.csv("Breast ap\_exp.csv", header = T)

dataNHS<- read.csv("NHS results.csv", header = T)

dataxNHS<- read.csv("NHS ap\_exp.csv", header = T)

dataNHIS<- read.csv("NHIS results.csv", header = T)

dataxNHIS<- read.csv("NHIS ap\_exp.csv", header = T)

dataCanCHEC1991<- read.csv("CanCHEC1991 results.csv", header = T)

dataxCanCHEC1991<- read.csv("CanCHEC1991 ap\_exp.csv", header = T)

dataCTS<- read.csv("CTS results.csv", header = T)

dataxCTS<- read.csv("CTS ap\_exp.csv", header = T)

dataVHM<- read.csv("VHM&PP results.csv", header = T)

dataxVHM<- read.csv("VHM&PP ap\_exp.csv", header = T)

dataDUELS <- read.csv(file = "DUELS results.csv", head=TRUE, sep=";", na.strings=c("."))

dataxDUELS <- read.csv(file = "DUELS ap\_exp.csv", head=TRUE, sep=";", na.strings=c("."))

#read in HR and CI for 18 ESCAPE cohorts not including VHM&PP cohort

dataESC<- read.csv("ESCAPE logHR se without VHM&PP.csv ", header = T)

#read in HR and CI for other cohorts

dataREST<- read.csv("HR CI Rest of World.csv ", header = T)

#calculate logHR and standard error based on 5th to 95th PM2.5 percentiles

logrESC=dataESC[,2]

seESC=dataESC[,3]

denESC=dataESC[,5]

numESC=dataESC[,6]

logrESC=logrESC\*(numESC-denESC)

seESC=seESC\*(numESC-denESC)

#convert HR based on 10ug/m3 contrast to logHR based on 5th to 95th percentiles and standard error

denREST=dataREST[,5]

numREST=dataREST[,6]

logrREST=(log(dataREST[,2])/10)\*(numREST-denREST)

seREST=((log(dataREST[,4])-log(dataREST[,3]))/(10\*2\*1.96))\*(numREST-denREST)

#for each of the 15 cohorts with subject level data, convert model results to ensemble prediction

#of logHR (and calculate maximum variance over concentration range assuming all predictions

#are perfectly correlated – this information is used in pooling curves

xx=dataxDUELS[,2]

nxDUELS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=18

mu=dataDUELS[a:e,2]

beta=dataDUELS[a:e,3]

se=dataDUELS[a:e,4]

tau=dataDUELS[a:e,5]

#ll=-dataDUELS[a:e,6]/2

f=as.character(dataDUELS[a:e,9])

weight=dataDUELS[a:e, 7]

output=cbind(f, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

#ll=as.numeric(out[,2])

wt=as.numeric(out[,2])

wt=matrix(wt)

mu=as.numeric(out[,3])

tau=as.numeric(out[,4])

beta=as.numeric(out[,5])

se=as.numeric(out[,6])

#wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VDUELS=diag(mvar, nx-1, nx-1)

MDUELS=rmean[2:nx]

DUELSden=xx[1]

DUELSnum=xx[2:nx]

rDUELS=mean(sd^2)/mean(sdw^2)

xx=dataxCTS[,2]

nxCTS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=19

mu=dataCTS[a:e,2]

beta=dataCTS[a:e,3]

se=dataCTS[a:e,4]

tau=dataCTS[a:e,5]

#ll=-dataCTS[a:e,6]/2

f=as.character(dataCTS[a:e,9])

weight=dataCTS[a:e, 7]

output=cbind(f, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

#ll=as.numeric(out[,2])

wt=as.numeric(out[,2])

wt=matrix(wt)

mu=as.numeric(out[,3])

tau=as.numeric(out[,4])

beta=as.numeric(out[,5])

se=as.numeric(out[,6])

#wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VCTS=diag(mvar, nx-1, nx-1)

MCTS=rmean[2:nx]

CTSden=xx[1]

CTSnum=xx[2:nx]

rCTS=mean(sd^2)/mean(sdw^2)

xx=dataxVHM[,2]

nxVHM=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=20

mu=dataVHM[a:e,2]

beta=dataVHM[a:e,3]

se=dataVHM[a:e,4]

tau=dataVHM[a:e,5]

#ll=-dataVHM[a:e,6]/2

f=as.character(dataVHM[a:e,9])

weight=dataVHM[a:e, 7]

output=cbind(f, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

#ll=as.numeric(out[,2])

wt=as.numeric(out[,2])

wt=matrix(wt)

mu=as.numeric(out[,3])

tau=as.numeric(out[,4])

beta=as.numeric(out[,5])

se=as.numeric(out[,6])

#wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VVHM=diag(mvar, nx-1, nx-1)

MVHM=rmean[2:nx]

VHMden=xx[1]

VHMnum=xx[2:nx]

rVHM=mean(sd^2)/mean(sdw^2)

xx=dataxBreast[,2]

nxBreast=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=19

mu=dataBreast[a:e,2]

beta=dataBreast[a:e,3]

se=dataBreast[a:e,4]

tau=dataBreast[a:e,5]

#ll=-dataBreast[a:e,6]/2

f=as.character(dataBreast[a:e,9])

weight=dataBreast[a:e, 7]

output=cbind(f, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

#ll=as.numeric(out[,2])

wt=as.numeric(out[,2])

wt=matrix(wt)

mu=as.numeric(out[,3])

tau=as.numeric(out[,4])

beta=as.numeric(out[,5])

se=as.numeric(out[,6])

#wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VBreast=diag(mvar, nx-1, nx-1)

MBreast=rmean[2:nx]

Breastden=xx[1]

Breastnum=xx[2:nx]

rBreast=mean(sd^2)/mean(sdw^2)

xx=dataxCanCHEC1991[,2]

nxCanCHEC1991=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=20

mu=dataCanCHEC1991[a:e,2]

beta=dataCanCHEC1991[a:e,3]

se=dataCanCHEC1991[a:e,4]

tau=dataCanCHEC1991[a:e,5]

ll=-dataCanCHEC1991[a:e,6]/2

f=as.character(dataCanCHEC1991[a:e,9])

weight=dataCanCHEC1991[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VCanCHEC1991=diag(mvar, nx-1, nx-1)

MCanCHEC1991=rmean[2:nx]

CanCHEC1991den=xx[1]

CanCHEC1991num=xx[2:nx]

rCanCHEC1991=mean(sd^2)/mean(sdw^2)

xx=dataxCanCHEC2001[,2]

nxCanCHEC2001=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=18

mu=dataCanCHEC2001[a:e,2]

beta=dataCanCHEC2001[a:e,3]

se=dataCanCHEC2001[a:e,4]

tau=dataCanCHEC2001[a:e,5]

ll=-dataCanCHEC2001[a:e,6]/2

f=as.character(dataCanCHEC2001[a:e,9])

weight=dataCanCHEC2001[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VCanCHEC2001=diag(mvar, nx-1, nx-1)

MCanCHEC2001=rmean[2:nx]

CanCHEC2001den=xx[1]

CanCHEC2001num=xx[2:nx]

rCanCHEC2001=mean(sd^2)/mean(sdw^2)

xx=dataxAARP[,2]

nxAARP=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=21

mu=dataAARP[a:e,4]

beta=dataAARP[a:e,6]

se=dataAARP[a:e,7]

tau=dataAARP[a:e,5]

ll=-dataAARP[a:e,8]/2

f=as.character(dataAARP[a:e,2])

weight=dataAARP[a:e, 9]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "linear"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VAARP=diag(mvar, nx-1, nx-1)

MAARP=rmean[2:nx]

AARPden=xx[1]

AARPnum=xx[2:nx]

rAARP=mean(sd^2)/mean(sdw^2)

xx=dataxCCHS[,2]

nxCCHS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=19

mu=dataCCHS[a:e,2]

beta=dataCCHS[a:e,3]

se=dataCCHS[a:e,4]

tau=dataCCHS[a:e,5]

ll=-dataCCHS[a:e,6]/2

f=as.character(dataCCHS[a:e,9])

weight=dataCCHS[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VCCHS=diag(mvar, nx-1, nx-1)

MCCHS=rmean[2:nx]

CCHSden=xx[1]

CCHSnum=xx[2:nx]

rCCHS=mean(sd^2)/mean(sdw^2)

xx=dataxUK[,2]

nxUK=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=19

mu=dataUK[a:e,4]

beta=dataUK[a:e,6]

se=dataUK[a:e,7]

tau=dataUK[a:e,5]

ll=-dataUK[a:e,8]/2

f=as.character(dataUK[a:e,2])

weight=dataUK[a:e, 9]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "linear"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VUK=diag(mvar, nx-1, nx-1)

MUK=rmean[2:nx]

UKden=xx[1]

UKnum=xx[2:nx]

rUK=mean(sd^2)/mean(sdw^2)

xx=dataxHK[,2]

nxHK=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=20

mu=dataHK[a:e,2]

beta=dataHK[a:e,3]

se=dataHK[a:e,4]

tau=dataHK[a:e,5]

ll=-dataHK[a:e,6]/2

f=as.character(dataHK[a:e,9])

weight=dataHK[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VHK=diag(mvar, nx-1, nx-1)

MHK=rmean[2:nx]

HKden=xx[1]

HKnum=xx[2:nx]

rHK=mean(sd^2)/mean(sdw^2)

xx=dataxACS[,2]

nxACS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=21

mu=dataACS[a:e,2]

beta=dataACS[a:e,3]

se=dataACS[a:e,4]

tau=dataACS[a:e,5]

ll=-dataACS[a:e,6]/2

f=as.character(dataACS[a:e,9])

weight=dataACS[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VACS=diag(mvar, nx-1, nx-1)

MACS=rmean[2:nx]

ACSden=xx[1]

ACSnum=xx[2:nx]

rACS=mean(sd^2)/mean(sdw^2)

xx=dataxChina[,2]

nxChina=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=18

mu=dataChinaH[a:e,2]

beta=dataChinaH[a:e,3]

se=dataChinaH[a:e,4]

tau=dataChinaH[a:e,5]

ll=-dataChinaH[a:e,6]/2

f=as.character(dataChinaH[a:e,9])

weight=dataChinaH[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

MChinaH=rmean[2:nx]

varH=sd^2

rChinaH=mean(sd^2)/mean(sdw^2)

xx=dataxChina[,2]

nxChina=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=18

mu=dataChinaL[a:e,2]

beta=dataChinaL[a:e,3]

se=dataChinaL[a:e,4]

tau=dataChinaL[a:e,5]

ll=-dataChinaL[a:e,6]/2

f=as.character(dataChinaL[a:e,9])

weight=dataChinaL[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

varL=sd^2

MChinaL=rmean[2:nx]

Chinaave=(MChinaH+MChinaL)/2

sdH=sqrt(varH[2:nx]+ (MChinaH-Chinaave)^2)

sdL=sqrt(varL[2:nx]+ (MChinaL-Chinaave)^2)

sdall=(sdL+sdH)/2

ss=matrix(0, nx-1, 1)

VV=(sdall)%\*%t(sdall)

for ( i in 1:nx-1) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VChina=diag(mvar, nx-1, nx-1)

MChinaL=rmean[2:nx]

Chinaden=xx[1]

Chinanum=xx[2:nx]

rChinaL=mean(sd^2)/mean(sdw^2)

MChina=(MChinaH+MChinaL)/2

rChina=(rChinaH+rChinaL)/2

xx=dataxNHS[,2]

nxNHS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=20

mu=dataNHS[a:e,2]

beta=dataNHS[a:e,3]

se=dataNHS[a:e,4]

tau=dataNHS[a:e,5]

ll=-dataNHS[a:e,6]/2

f=as.character(dataNHS[a:e,9])

weight=dataNHS[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VNHS=diag(mvar, nx-1, nx-1)

MNHS=rmean[2:nx]

NHSden=xx[1]

NHSnum=xx[2:nx]

rNHS=mean(sd^2)/mean(sdw^2)

xx=dataxNHIS[,2]

nxNHIS=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=19

mu=dataNHIS[a:e,2]

beta=dataNHIS[a:e,3]

se=dataNHIS[a:e,4]

tau=dataNHIS[a:e,5]

ll=-dataNHIS[a:e,6]/2

f=as.character(dataNHIS[a:e,9])

weight=dataNHIS[a:e, 7]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

wt=exp((ll-max(ll)))/sum(exp(((ll-max(ll)))))

wt=matrix(wt)

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "z\*logit"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log(z)\*logit"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VNHIS=diag(mvar, nx-1, nx-1)

MNHIS=rmean[2:nx]

NHISden=xx[1]

NHISnum=xx[2:nx]

rNHIS=mean(sd^2)/mean(sdw^2)

xx=dataxRome[,2]

nxRome=length(xx)-1

x=xx-min(xx)

nx=length(x)

r=max(x)-min(x)

a=1

e=18

mu=dataRome[a:e,4]

beta=dataRome[a:e,6]

se=dataRome[a:e,7]

tau=dataRome[a:e,5]

ll=-dataRome[a:e,8]/2

f=as.character(dataRome[a:e,2])

weight=dataRome[a:e, 9]

output=cbind(f, ll, weight, mu, tau, beta, se)

out=subset(output, weight>0)

f=out[,1]

ll=as.numeric(out[,2])

wt=as.numeric(out[,3])

wt=matrix(wt)

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

nf=length(f)

T=matrix(0, nx, nf)

sd=matrix(0, nx, 1)

sdw=matrix(0, nx, 1)

sdb=matrix(0, nx, 1)

rmean=matrix(0, nx, 1)

meanrisk=matrix(0, nx, 1)

var=matrix(0,nx,nf)

varw=matrix(0,nx,nf)

varb=matrix(0,nx,nf)

upcl<-matrix(0, nx, 1)

lowcl<-matrix(0, nx, 1)

for (i in 1:nx) {

for (k in 1:nf) {

if (f[k]== "linear"){

T[i,k]<-x[i]/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

if (f[k] == "log"){

T[i,k]<-log(x[i]+1)/(1+exp(-(x[i]-mu[k])/(tau[k]\*r)))

}

}}

for (i in 1:nx) {rmean[i]=(T[i,]\*beta)%\*%wt}

for (k in 1:nf) { for (i in 1:nx) {

var[i,k]=(T[i,k]\*se[k])^2 + (T[i,k]\*beta[k]-rmean[i])^2

varw[i,k]=(T[i,k]\*se[k])^2

varb[i,k]=(T[i,k]\*beta[k]-rmean[i])^2 }}

for (i in 1:nx) {

sd[i]=sqrt(var[i,])%\*%wt

sdw[i]=sqrt(varw[i,])%\*%wt

sdb[i]=sqrt(varb[i,])%\*%wt

}

ss=matrix(0, nx, 1)

VV=(sd)%\*%t(sd)

for ( i in 1:nx) { ss[i]=sum(VV[i,])}

mvar= max(ss)

VRome=diag(mvar, nx-1, nx-1)

MRome=rmean[2:nx]

Romeden=xx[1]

Romenum=xx[2:nx]

rRome=mean(sd^2)/mean(sdw^2)

#calculate mean of total to sampling error among 15 cohorts

ratio=c(rBreast, rCanCHEC1991, rACS, rNHS, rNHIS, rHK, rUK, rAARP, rRome, rCCHS, rCanCHEC2001, rVHM, rChina, rDUELS, rCTS)

infl=mean(ratio)

vESC=seESC^2

VESC=bdiag(vESC[1], vESC[2], vESC[3], vESC[4], vESC[5], vESC[6], vESC[7], vESC[8], vESC[9], vESC[10], vESC[11], vESC[12], vESC[13], vESC[14], vESC[15], vESC[16], vESC[17], vESC[18])\*infl

vREST=seREST^2

VREST=vREST\*infl

VESC=VESC\*infl

VRESTUS=bdiag(vREST[1], vREST[2], vREST[3], vREST[5], vREST[8])

VUS=bdiag(VACS, VAARP, VNHS, VNHIS, VCTS, VRESTUS)

VRESTEur=bdiag(vREST[4], vREST[7])

VEurope=bdiag(VUK, VRome, VDUELS, VVHM, VESC, VRESTEur)

VRESTASIA=bdiag(VREST[6])

VAsia=bdiag(VHK, VChina, VRESTASIA)

VCanada=bdiag(VCanCHEC1991, VCanCHEC2001, VCCHS, VBreast)

Vall=bdiag(VCanada, VUS, VEurope, VAsia)

logr=c(MCanCHEC1991, MCanCHEC2001, MCCHS, MBreast, MACS, MAARP, MNHS, MNHIS, MCTS, logrREST[1], logrREST[2], logrREST[3], logrREST[5], logrREST[8], MUK, MRome, MDUELS, MVHM, logrESC, logrREST[4], logrREST[7], MHK, MChina, logrREST[6])

den=c(rep(CanCHEC1991den, nxCanCHEC1991), rep(CanCHEC2001den, nxCanCHEC2001), rep(CCHSden, nxCCHS), rep(Breastden, nxBreast), rep(ACSden, nxACS), rep(AARPden, nxAARP), rep(NHSden, nxNHS), rep(NHISden, nxNHIS), rep(CTSden, nxCTS), denREST[1], denREST[2], denREST[3], denREST[5], denREST[8], rep(UKden, nxUK), rep(Romeden, nxRome), rep(DUELSden, nxDUELS), rep(VHMden, nxVHM), denESC, denREST[4], denREST[7], rep(HKden, nxHK), rep(Chinaden, nxChina), denREST[6])

num=c(CanCHEC1991num, CanCHEC2001num, CCHSnum, Breastnum, ACSnum, AARPnum, NHSnum, NHISnum, CTSnum, numREST[1], numREST[2], numREST[3], numREST[5], numREST[8], UKnum, Romenum, DUELSnum, VHMnum, numESC, numREST[4], numREST[7], HKnum, Chinanum, numREST[6])

study=c(rep(1, nxCanCHEC1991), rep(2, nxCanCHEC2001), rep(3, nxCCHS), rep(4, nxBreast), rep(5, nxACS), rep(6, nxAARP), rep(7, nxNHS), rep(8, nxNHIS), rep(9, nxCTS), 10, 11, 12, 13, 14, rep(15, nxUK), rep(16, nxRome), rep(17, nxDUELS), rep(18, nxVHM), seq(19, 36), 37, 38, rep(39, nxHK), rep(40, nxChina), 41)

study=as.factor(study)

nxCanada=nxCanCHEC1991+nxCanCHEC2001+nxCCHS+nxBreast

nxUS=nxACS+nxAARP+nxNHS+nxNHIS+nxCTS+5

nxEurope=nxUK+nxRome+nxDUELS+nxVHM+length(logrESC)+2

nxAsia=nxHK+nxChina+1

region=c(rep(1, nxCanada), rep(2, nxUS), rep(3, nxEurope), rep(4, nxAsia))

region=as.factor(region)

maxnum=c(max(CTSnum), max(Breastnum), max(AARPnum), max(CCHSnum), max(ACSnum), max(UKnum), max(HKnum), max(NHSnum), max(NHISnum), max(CanCHEC1991num), max(CanCHEC2001num), max(VHMnum), max(Romenum), max(Chinanum), max(DUELSnum), numESC, numREST)

minden=c(CTSden, Breastden, AARPden, CCHSden, ACSden, UKden, HKden, NHSden, NHISden, CanCHEC1991den, CanCHEC2001den, VHMden, Romeden, Chinaden, DUELSden, denESC, denREST)

THRES=min(minden)

r=max(num)-min(den)

a=c(1,3,5,7,9)

e=c(minden, maxnum)-THRES

mm0=min(e)

mm25=quantile(e, 0.25)

mm50=quantile(e, 0.5)

mm75=quantile(e, 0.75)

mm95=quantile(e, 0.95)

m=c(mm0, mm25, mm50, mm75, mm95)

max1=length(m)\*length(a)

max2=max1

max3=max1

aic1<-matrix(0, max1, 1)

b1<-matrix(0, max1, 1)

se1<-matrix(0, max1, 1)

model.form1<-matrix(0, max1, 1)

set.tau1<-matrix(0, max1, 1)

mu1<-matrix(0, max1, 1)

aic2<-matrix(0, max2, 1)

b2<-matrix(0, max2, 1)

se2<-matrix(0, max2, 1)

model.form2<-matrix(0, max2, 1)

set.tau2<-matrix(0, max2, 1)

mu2<-matrix(0, max2, 1)

aic3<-matrix(0, max3, 1)

b3<-matrix(0, max3, 1)

se3<-matrix(0, max3, 1)

model.form3<-matrix(0, max3, 1)

set.tau3<-matrix(0, max3, 1)

mu3<-matrix(0, max3, 1)

aic4<-matrix(0, max1, 1)

b4<-matrix(0, max1, 1)

se4<-matrix(0, max1, 1)

model.form4<-matrix(0, max1, 1)

set.tau4<-matrix(0, max1, 1)

mu4<-matrix(0, max1, 1)

aic5<-matrix(0, max1, 1)

b5<-matrix(0, max1, 1)

se5<-matrix(0, max1, 1)

model.form5<-matrix(0, max1, 1)

set.tau5<-matrix(0, max1, 1)

mu5<-matrix(0, max1, 1)

aic6<-matrix(0, max1, 1)

b6<-matrix(0, max1, 1)

se6<-matrix(0, max1, 1)

model.form6<-matrix(0, max1, 1)

set.tau6<-matrix(0, max1, 1)

mu6<-matrix(0, max1, 1)

numt=((num-THRES) + abs(num-THRES))/2

dent=((den-THRES) + abs(den-THRES))/2

#estimate theta using rma.mv by defining difference in shapes between num and den #concentrations

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.1\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.1\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic1[j]=AIC(fit)

b1[j]=coef(fit)

model.form1[j]<- a[k]

set.tau1[j] <- 0.1

mu1[j] <- m[i]

se1[j]=sqrt(vcov(fit))}}

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.2\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.2\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic2[j]=AIC(fit)

b2[j]=coef(fit)

model.form2[j]<- a[k]

set.tau2[j] <- 0.2

mu2[j] <- m[i]

se2[j]=sqrt(vcov(fit))}}

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.3\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.3\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic3[j]=AIC(fit)

b3[j]=coef(fit)

model.form3[j]<- a[k]

set.tau3[j] <- 0.3

mu3[j] <- m[i]

se3[j]=sqrt(vcov(fit))}}

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.4\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.4\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic4[j]=AIC(fit)

b4[j]=coef(fit)

model.form4[j]<- a[k]

set.tau4[j] <- 0.4

mu4[j] <- m[i]

se4[j]=sqrt(vcov(fit))}}

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.5\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.5\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic5[j]=AIC(fit)

b5[j]=coef(fit)

model.form5[j]<- a[k]

set.tau5[j] <- 0.5

mu5[j] <- m[i]

se5[j]=sqrt(vcov(fit))}}

j=0

for (k in 1:length(a)) {

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

j=j+1

diff=log(numt/a[k]+1)/(1+exp(-(numt-m[i])/(0.6\*r))) - log(dent/a[k]+1)/(1+exp(-(dent-m[i])/(0.6\*r)))

fit=rma.mv(yi=logr, V=Vall, mods=~diff -1, random=list( ~ 1 | study), method="REML", intercept=FALSE,

struct="CS")

aic6[j]=AIC(fit)

b6[j]=coef(fit)

model.form6[j]<- a[k]

set.tau6[j] <- 0.6

mu6[j] <- m[i]

se6[j]=sqrt(vcov(fit))}}

#complie necessary information from model fitting to construct ensemble estimate and bootstrap

#based CI

aic=rbind(aic1, aic2, aic3, aic4, aic5, aic6)

b=rbind( b1, b2, b3, b4, b5, b6)

se=rbind(se1, se2, se3, se4, se5, se6)

model.form <- rbind(model.form1, model.form2, model.form3, model.form4, model.form5, model.form6)

model.form=as.character(model.form)

set.tau <- rbind( set.tau1, set.tau2, set.tau3, set.tau4, set.tau5, set.tau6)

mu <- rbind(mu1, mu2, mu3, mu4, mu5, mu6)

wt=exp(-0.5\*(aic-min(aic)))/sum(exp(-0.5\*(aic-min(aic))))

output=cbind(model.form, aic, wt, mu, set.tau, b, se)

out=subset(output, wt>0)

f=out[,1]

aic=as.numeric(out[,2])

wt=as.numeric(out[,3])

mu=as.numeric(out[,4])

tau=as.numeric(out[,5])

beta=as.numeric(out[,6])

se=as.numeric(out[,7])

weight=exp(-0.5\*(aic-min(aic)))/sum(exp(-0.5\*(aic-min(aic))))

expo\_name <- c("PM2.5")

expo\_unit <- c("ug/m3")

nn <- length(f)

finalmodels.best.nn.final <- data.frame(tau=tau, funcform=f, coef=beta, se.coef=se, wt.final3=weight, loca\_perc=mu)

MM=max(num)

MM=84

bb=1

x=seq(0, MM, by=bb)

nx=length(x)

#run routine to construct enesembe estimate and bootstrap CI

risk <- function(ap\_data, finalmodels, expo\_name, unit, nn, TT){

x=seq(0, MM, by=bb)

# prepare x1 for use in simulation, varying depending on perc, set\_tau, and funcform

sim.x1 <- function(x\_sim, perc, set\_tau\_sim, funcform\_sim, TT){

mu <- perc

#r <- max(num)-min(den)

thr=((x\_sim-TT)+abs(x\_sim-TT))/2

if (funcform\_sim=="1"){x1<- log(thr+1)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="2"){x1<-log(1+ thr/2)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="3"){x1<-log(1+ thr/3)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="4"){x1<-log(1+ thr/4)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="5"){x1<-log(1+ thr/5)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="6"){x1<-log(1+ thr/6)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="7"){x1<-log(1+ thr/7)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="8"){x1<-log(1+ thr/8)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="9"){x1<-log(1+ thr/9)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="10"){x1<-log(1+ thr/10)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="11"){x1<-log(1+ thr/11)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="12"){x1<-log(1+ thr/12)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="13.8"){x1<-log(1+ thr/13.8)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="14"){x1<-log(1+ thr/14)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="15"){x1<-log(1+ thr/15)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="16"){x1<-log(1+ thr/16)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="17"){x1<-log(1+ thr/17)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="18"){x1<-log(1+ thr/18)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="19.9"){x1<-log(1+ thr/19.9)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="20"){x1<-log(1+ thr/20)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="21"){x1<-log(1+ thr/21)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="22"){x1<-log(1+ thr/22)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="23"){x1<-log(1+ thr/23)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="24"){x1<-log(1+ thr/24)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="25"){x1<-log(1+ thr/25)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="26"){x1<-log(1+ thr/26)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="27"){x1<-log(1+ thr/27)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="28"){x1<-log(1+ thr/28)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="29"){x1<-log(1+ thr/29)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="30"){x1<-log(1+ thr/30)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="31"){x1<-log(1+ thr/31)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="32"){x1<-log(1+ thr/32)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="33"){x1<-log(1+ thr/33)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="34"){x1<-log(1+ thr/34)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="35"){x1<-log(1+ thr/35)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="36"){x1<-log(1+ thr/36)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="37"){x1<-log(1+ thr/37)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="38"){x1<-log(1+ thr/38)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="39"){x1<-log(1+ thr/39)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="40"){x1<-log(1+ thr/40)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="41"){x1<-log(1+ thr/41)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="42"){x1<-log(1+ thr/42)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="43"){x1<-log(1+ thr/43)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="44"){x1<-log(1+ thr/44)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="45"){x1<-log(1+ thr/45)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="46"){x1<-log(1+ thr/46)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="47"){x1<-log(1+ thr/47)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="48"){x1<-log(1+ thr/48)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="49"){x1<-log(1+ thr/49)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="50"){x1<-log(1+ thr/50)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

if (funcform\_sim=="81"){x1<-log(1+ thr/81)/(1+exp(-(thr-mu)/(set\_tau\_sim\*r)))}

return(x1)

}

# simulate 100,000 realizations based on se.coef AND weights derived from LL

if (nn >= 2) {

nsim<-10000

ran.3<-matrix(0, nsim, 1)

rr.3<-matrix(0, nsim, nx)

medRR.3<-matrix(0, nx, 1)

upcl.3<-matrix(0, nx, 1)

lowcl.3<-matrix(0, nx, 1)

pp <- 1 # position variable in the 100,000 sim

nn <- nn # consider top 3 models

nsim.sum <- 0 # count of nsim to ensure the last run lead to rownum of exactly 100,000

# k from 0 to nn-1: varying depending on models included and pooled

for (k in 0:(nn-1)) {

nsim.wt <- nsim \* round(finalmodels[length(finalmodels[,1])-k,]$wt.final3, digits =4)

loca\_perc\_sim <- finalmodels[length(finalmodels[,1])-k,]$loca\_perc

funcform\_sim <- finalmodels[length(finalmodels[,1])-k,]$funcform

set\_tau <- finalmodels[length(finalmodels[,1])-k,]$tau

x1 <- sim.x1(x\_sim=x, perc=loca\_perc\_sim, set\_tau\_sim=set\_tau, funcform\_sim=funcform\_sim, TT=THRES)

if (k==nn-1) {nsim.wt <- nsim - nsim.sum}

for (i in pp:(pp+nsim.wt-1)) {

if (i<=10000){

ran.3[i,]<-rnorm(1, finalmodels[length(finalmodels[,1])-k,]$coef, finalmodels[length(finalmodels[,1])-k,]$se.coef)

for (j in 1:length(x)){

rr.3[i,j]<-exp(ran.3[i,1]\*x1[j])

}

}

}

pp <- pp+nsim.wt

nsim.sum <- nsim.sum+nsim.wt

}

return(rr.3)

}

}

rr.3=risk(ap\_data=x, nn=nn, finalmodels=finalmodels.best.nn.final, expo\_name=expo\_name, unit=expo\_unit, TT=THRES)

mean=matrix(0, length(x), 1)

ucl=matrix(0, length(x), 1)

lcl=matrix(0, length(x), 1)

sd=matrix(0, length(x), 1)

for (j in 1:length(x)) {

mean[j]=mean(rr.3[,j])

ucl[j]=quantile(rr.3[,j], 0.975)

lcl[j]=quantile(rr.3[,j], 0.025)

sd[j]=sd(log(rr.3[,j]))

}

#estimate approximate function to ensemble estimate and assign all uncertainty to theta

xxx=((x-THRES)+abs(x-THRES))/2

logmean=log(mean)

taustart=0.4\*r

fitmean=nlsLM(logmean~b\*log(xxx/mTT+1)/(1+exp(-(xxx-mu)/tau)), start=list(b=0.1, mu=10, tau=taustart, mTT=5))

mb=coef(fitmean)[1]

mmu=coef(fitmean)[2]

mt=coef(fitmean)[3]

mTT=coef(fitmean)[4]

fitsd=glm(sd~logmean -1 )

sdapprox=mb\*coef(fitsd)

meanr=exp(mb\*log(xxx/mTT+1)/(1+exp(-(xxx-mmu)/mt)))

lclr=exp((mb-1.96\*sdapprox)\*log(xxx/mTT+1)/(1+exp(-(xxx-mmu)/mt)))

uclr=exp((mb+1.96\*sdapprox)\*log(xxx/mTT+1)/(1+exp(-(xxx-mmu)/mt)))

#plot approx GEMM and CI

x=seq(0, MM, by=bb)

plot(x, uclr, lwd=4, type="l", col="lightgrey", ylab="Relative Risk", xlab= expression(paste("PM"[2.5], " - ", mu, "g/m"^3)))

polygon(x=c(x, rev(x)), y=c(lclr, rev(uclr)), col="lightgrey", lty=2, border=NA)

lines(x, meanr, lwd=4, col="red")

abline(1,0)

cbind(mb, sdapprox, mTT, mmu, mt)