**R CODES FOR EMPIRICAL BAYES AND BAYESIAN ANALYSES OF MULTIPLE RISK FACTORS IN MATCHED CASE-CONTROL STUDIES.**

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

#

# REFERENCE:

#

# JM Satagopan, A Sen, Q Zhou, Q Lan, N Rothman, H Langseth, LS Engel (2015).

# Bayes and empirical Bayes methods for reduced rank regression models in

# matched case-control studies. Biometrics (In Press). PMID: 26575519.

#

# This file contains:

# 1. R codes for empirical Bayes analyses;

# 2. BUGS codes for Bayesian analyses; and

# 3. R codes to call the BUGS codes from Linux using the rjags library.

#

#

# Author:

# Jaya M. Satagopan (satagopj@mskcc.org)

#

# October, 2015

#

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

**R CODE FOR EMPIRICAL BAYES ANALYSIS**

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

#

#

# EB analysis

#

# The input file "whole.data" is a matrix containing the data set.

# It has 2N rows (N cases and N matched controls).

# The columns are the risk factors and the outcome

#

# In particular, the column "y.name" contains the binary case/control status.

#

# "pcb.name" contains the column names of the PCBs to be analyzed.

#

# "match.pair.name" contains the pair indicator.

# For example, the first pair will contain the same code

# (for example, the character "ID.1")

# for the case and control in that pair.

#

# "weight.matrix" is the weight matrix.

# It contains number of rows equal to the length of "pcb.name".

# The row names must be the same as

# "pcb.name".

#

# "factor.name" contains the names of factors for adjustment in the analysis.

#

#

# EXAMPLE RUN:

#

# nhl.result<-pcb\_factor\_EB\_func(whole.data=input.data, y.name="casectrl",

# pcb.name=simu.pcb.list, match.pair.name="pairno",

# weight.matrix=weight1.mat, # factor.name=c("bmi\_median","bmi\_high",

# "smoke\_former","smoke\_current"))

#

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

pcb\_factor\_EB\_func<- function(whole.data, y.name, pcb.name, match.pair.name,weight.matrix,factor.name){

 if(!is.vector(weight.matrix)){ pcb.name<-pcb.name[match(rownames(weight.matrix), pcb.name) ] }

 else { pcb.name<-pcb.name[match(names(weight.matrix), pcb.name) ]

 weight.matrix<-matrix(weight.matrix,ncol=1,byrow=T) }

 x.pcb.pre<-whole.data[,names(whole.data) %in% pcb.name]

 x.pcb<-x.pcb.pre[,match(pcb.name,names(x.pcb.pre))]

 x.pcb<-as.matrix(x.pcb)

 p.numb<-ncol(x.pcb)

 XW.x<-x.pcb %\*% weight.matrix

 ### Fit two simple conditional logistic regression and obtain the initial values first for future use

 common.col.name<-c(y.name,match.pair.name,factor.name)

 common.info.vec<-whole.data[,names(whole.data) %in% common.col.name]

 common.vec.add<-common.info.vec[,match(common.col.name,names(common.info.vec))]

 factor.length<-length(factor.name)

 pcb.ele.start<-factor.length+1

 m1.data<-data.frame(common.vec.add,x.pcb)

 m2.data<-data.frame(common.vec.add,XW.x)

 m1.data.xmat<-m1.data[,-c(1,2)]

 m2.data.xmat<-as.data.frame(m2.data[,-c(1,2)])

 library(survival)

###

# fit full model (fit\_1) and reduced model (fit\_2)

###

 fit\_1<-clogit(m1.data[,1]~.+strata(m1.data[,2]), data=m1.data.xmat)

 fit\_2<-clogit(m2.data[,1]~.+strata(m2.data[,2]), data=m2.data.xmat)

 # print("Ready for calculation....")

 #### (1) Record the basic values from the upper two fits for MLE and RES

####

# extract estimates and variance of mle $b\_{full}$ from full model

####

 end.coef.mle<-length(coef(fit\_1))

 b\_mle<-coef(fit\_1)[pcb.ele.start:end.coef.mle]

 V\_mle<-V\_m1<-vcov(fit\_1)[pcb.ele.start:end.coef.mle,pcb.ele.start:end.coef.mle]

 b\_mle\_factor<-coef(fit\_1)[1:factor.length]

 V\_mle\_factor<-vcov(fit\_1)[1:factor.length, 1:factor.length]

####

# extract estimates and variance of mle d from reduced model

####

 end.coef.red<-length(coef(fit\_2))

 d\_red = coef(fit\_2)[pcb.ele.start:end.coef.red]

 V\_d <- vcov(fit\_2)[pcb.ele.start:end.coef.red, pcb.ele.start:end.coef.red]

 b\_red\_factor<-coef(fit\_2)[1:factor.length]

 V\_red\_factor<-vcov(fit\_2)[1:factor.length, 1:factor.length]

####

# use $d\_{mle}$ to calculate $b\_{red}$ for the reduced model and its variance

####

 b\_red <- weight.matrix %\*% d\_red

 V\_red <- weight.matrix %\*% V\_d %\*% t(weight.matrix)

 r\_val = as.vector(b\_mle-b\_red)

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

#

# calculating V\_r = Var(r)

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

 ### redefine the m1.data.xmat and m2.data.xmat for just keep the 36PCB

rm.length<-(factor.length+2)

 m1.data.xmat<-m1.data[,-seq(1,rm.length)]

 m2.data.xmat<-as.data.frame(m2.data[,-seq(1,rm.length)])

 ## start the old code

n1 <- ncol(m1.data.xmat)

n2 <- ncol(m2.data.xmat)

ufull.uredp.sum <- matrix(0, nrow=n1, ncol=n2)

ured.ufullp.sum <- matrix(0, nrow=n2, ncol=n1)

match.pair.id.vector <- whole.data[,match.pair.name]

unique.match.pair.id.vector <- unique(match.pair.id.vector)

nsamp <- nrow(m1.data)/2

for(i.temp in 1:nsamp){

 temp.id <- which(match.pair.id.vector == unique.match.pair.id.vector[i.temp])

 temp.y <- m1.data[temp.id, 1]

 temp.x <- m1.data[temp.id,-(1:rm.length)]

 temp.mu <- exp(fit\_1$linear.predictors[temp.id]) / sum( exp(fit\_1$linear.predictors[temp.id]))

 ufull.vector <- as.vector(temp.x[1,]) \* (temp.y[1] - temp.mu[1]) +

 as.vector(temp.x[2,]) \* (temp.y[2] - temp.mu[2])

 temp.x.red <- as.data.frame(m2.data[temp.id, -(1:rm.length)])

 temp.mu.red <- exp(fit\_2$linear.predictors[temp.id]) / sum(exp(fit\_2$linear.predictors[temp.id]))

 ured.vector <- as.vector(temp.x.red[1,]) \* (temp.y[1] - temp.mu.red[1]) +

 as.vector(temp.x.red[2,]) \* (temp.y[2] - temp.mu.red[2])

 ufull.mat <- matrix(as.double(ufull.vector), ncol=1)

 ured.mat <- matrix(as.double(ured.vector), ncol=1)

 ufull.uredp.sum <- ufull.uredp.sum + ufull.mat %\*% t(ured.mat)

 ured.ufullp.sum <- ured.ufullp.sum + ured.mat %\*% t(ufull.mat)

}

cov.bfull.bred <- V\_mle %\*% ufull.uredp.sum %\*% V\_d %\*% t(weight.matrix)

cov.bred.bfull <- weight.matrix %\*% V\_d %\*% ured.ufullp.sum %\*% V\_mle

 V\_r <- sigma <- V\_mle + V\_red - cov.bfull.bred - cov.bred.bfull

### V\_r <- sigma <- V\_mle - V\_red

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

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

####

# EB.2

# calculate the eb-type shrinkage estimate based on N(0, sigma^2 \* I) prior,

# denoted b\_eb.2

####

 rt\_r= as.numeric( t(r\_val) %\*% r\_val )

 I\_mat<-diag(p.numb)

 bs\_term1<-sigma+ (rt\_r/p.numb)\*I\_mat

 bs\_term2<-solve(bs\_term1)

 bs\_term3<-sigma %\*% bs\_term2 %\*% r\_val

 b\_eb.2 <- b\_mle-bs\_term3

####

# calculate the variance of b\_eb.2

####

 V.vector <- solve(sigma + rt\_r/p.numb \* I\_mat) %\*% r\_val

 r.repeated.matrix <- NULL

 for(i.numb in 1:p.numb){

 r.repeated.matrix <- rbind(r.repeated.matrix, r\_val)

 }

 V.vector <- as.vector(V.vector)

 V.diag.matrix <- diag(V.vector)

 C.matrix <- V.diag.matrix %\*% r.repeated.matrix

 M2.matrix <- sigma %\*% solve(sigma + rt\_r/p.numb \* I\_mat) %\*% (I\_mat - 2/p.numb \* C.matrix)

 M.matrix <- cbind(I\_mat - M2.matrix, M2.matrix)

 R.matrix.1 <- cbind(V\_mle, cov.bfull.bred)

 R.matrix.2 <- cbind(cov.bred.bfull, V\_red)

 R.matrix <- rbind(R.matrix.1, R.matrix.2)

 V\_eb.2 <- M.matrix %\*% R.matrix %\*% t(M.matrix)

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

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

####

# EB.1

# calculate the eb-type shrinkage EB.1 estimate based on exchangeable prior,

# denoted as b\_eb.1

####

 r\_rt= r\_val %\*% t(r\_val)

 A.matrix.eb1=r\_rt

 b\_eb.1 <- b\_mle - sigma %\*% solve(sigma + A.matrix.eb1) %\*% r\_val

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

#### VARIANCE OF EB.1 #######

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

My.M.Mat <- sigma %\*% solve(sigma + A.matrix.eb1)

I.minus.My.M.Mat <- diag(nrow(My.M.Mat)) - My.M.Mat

Var.EB.1.Taylor <- NULL

sigma.inv <- solve(sigma)

quad.form <- as.vector(t(r\_val) %\*% sigma.inv %\*% r\_val)

Var.Mat.1 <- 1/(1 + quad.form) \* diag(nrow(sigma)) -

 2 \* r\_val %\*% t(r\_val) %\*% sigma.inv / (1 + quad.form)^2

R.mat <- rbind( cbind(V\_mle, cov.bfull.bred), cbind(cov.bred.bfull, V\_red) )

pre.multiply.mat <- cbind(diag(nrow(Var.Mat.1))-Var.Mat.1, Var.Mat.1)

Var.EB.1.Taylor <- pre.multiply.mat %\*% R.mat %\*% t(pre.multiply.mat)

################# Organize the output

 estimate <- cbind(b\_mle, b\_red, b\_eb.2, b\_eb.1)

 colnames(estimate) <- c("MLE\_est","Res\_est", "EB.2\_est", "EB.1\_est")

 variance<-cbind(diag(V\_mle),diag(V\_red),diag(V\_eb.2),diag(Var.EB.1.Taylor))

 colnames(variance)<- c("MLE\_var","Res\_var","EB.2\_var", "EB.1.var.Taylor")

 factor.result<-cbind(b\_mle\_factor, diag(V\_mle\_factor), b\_red\_factor, diag(V\_red\_factor))

 colnames(factor.result)<-c("MLE\_est","MLE\_var","Red\_est","Red\_var")

 return(list(estimate=estimate, variance=variance,lambda=lambda,factor.result=factor.result, sigma=sigma, r.val=r\_val) )

 }

#! End of the pcb\_factor\_EB\_func

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

**BUGS CODE FOR BAYESIAN RIDGE (save this in file “bayesian-ridge.bug”)**

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

#

# September 3, 2014

#

# Bayesian Ridge bugs code

#

# beta are the beta parameters for the 36 PCBs

# beta\_other are the effects of the adjustment variables

#

# x\_case is a matrix of case data with pcbs in the first 36 columns and adjustment variables in the final 4 columns

# x\_control is a matrix of control data, organized in a manner similar to x\_case

#

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

model

 {

 for(i in 1:n){

 ##obtain the beta\*X value and the e[i,1] is the case value and e[i, 2] is the control value

 ## note: 1=case; 2=control

 log(e[i,1]) <- inprod(x\_case[i, 1:36],beta[1:36]) + inprod(x\_case[i, 37:40], beta\_other[1:4])

 log(e[i,2]) <- inprod(x\_ctrl[i, 1:36],beta[ 1:36]) + inprod(x\_ctrl[i, 37:40],beta\_other[1:4])

 Y[i,1]<-1

 Y[i,2]<-0

 ## conditional likelihood

 for(j in 1:2){

 p[i, j]<- e[i, j]/sum(e[i, ]) }

 Y[i, 1:2] ~ dmulti(p[i, 1:2], 1) }

 ## priors are defined below

########### we have 40 beta parameters in the real data,

# 36 are for PCB, 2 for BMI, 2 for Smoking

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

####### these 36 betas are for PCB ##########

 for(k in 1:36){

 mu[k] <- inprod(weight\_mat[k,1:4], d[1:4])

 beta[k]~dnorm(mu[k], tau\_b\_val)

 }

 tau\_b\_val ~ dgamma(2,2)

 t\_d\_val ~ dgamma(2, 2)

 for(g in 1:4){

 d[g]~dnorm(0, t\_d\_val)

 }

########## These 4 beta\_other are for bmi and smoking ##############

 t\_other\_val ~ dgamma(2,2)

 for(i in 1:4){

 beta\_other ~ dnorm(0, t\_other\_val)

 }

}

**BUGS CODE FOR BAYESIAN LASSO (save this in file “bayesian-lasso.bug”)**

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

#

# September 3, 2014

#

# Bayesian LASSO bugs code

#

# beta are the beta parameters for the 36 PCBs

# beta\_other are the effects of the adjustment variables

#

# x\_case is a matrix of case data with pcbs in the first 36 columns and adjustment variables in the final 4 columns

# x\_control is a matrix of control data, organized in a manner similar to x\_case

#

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

model

 {

 for(i in 1:n){

 ##obtain the beta\*X value and the e[i,1] is the case value and e[i, 2] is the control value

 ## note: 1=case; 2=control

 log(e[i,1]) <- inprod(x\_case[i, 1:36],beta[1:36]) + inprod(x\_case[i, 37:40], beta\_other[1:4])

 log(e[i,2]) <- inprod(x\_ctrl[i, 1:36],beta[ 1:36]) + inprod(x\_ctrl[i, 37:40],beta\_other[1:4])

 Y[i,1]<-1

 Y[i,2]<-0

 ## conditional likelihood

 for(j in 1:2){

 p[i, j]<- e[i, j]/sum(e[i, ]) }

 Y[i, 1:2] ~ dmulti(p[i, 1:2], 1) }

 ## priors are defined below

####### these 36 betas are for PCB ##########

 for(k in 1:36){

 mu[k] <- inprod(weight\_mat[k,1:4], d[1:4])

 beta[k]~dnorm(mu[k], tau\_b[k])

 sigma2\_b[k] <- sigma2\_val \* tg2[k]

 tg2[k] ~ dexp(temp\_lamb)

 tau\_b[k] <- 1/(sigma2\_b[k])

 }

 t\_b~dgamma(1, 1)

 sigma2\_val <-(1/t\_b)

 temp\_lamb<-(lambda\*lambda)/2

 for(g in 1:4){

 d[g]~dnorm(0, t\_b)

 }

#######

## lambda.squared has a gamma prior

#######

 lambda <- sqrt(lambda.2)

 lambda.2 ~ dgamma(5, 1)

########## These 4 beta\_other are for bmi and smoking ##############

 t\_other\_val ~ dgamma(2,2)

 for(i in 1:4){

 beta\_other ~ dnorm(0, t\_other\_val)

 }

}

**R FUNCTION TO CALL THE BUGS CODES THROUGH THE RJAGS LIBRARY**

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

#

# September 3, 2014

#

# COMMANDS TO RUN MCMC USING RJAGS BY CALLING THE BUGS CODES FOR BAYESIAN RIDGE AND BAYESIAN LASSO

#

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

library(rjags)

winbug.func <- "bayesian-ridge.bug"

# bayesian-ridge.bug contains bugs code for running bayesian ridge

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

# uncomment the following command, and comment the above command to run bayesian lasso

#

#winbug.func <- "bayesia-lasso.bug"

## bayesian-lasso.bug contains bugs code for running bayesian lasso

#

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

 n=190 ##### number of case-control.pairs

 my.p=36 ##### number of pcbs

 my.other=4 ##### number of adjustment variables

 my.d=4 ##### number ofm columns of weight matrix.

 Y <- cbind(rep(1,n), rep(0,n)) ######### case-control status

#### case data

 temp\_case <- read.table("x1.txt", header=T) ######### in a file "x1.txt", store the case data with n=190 rows and my.p+my.other = 40 columns

 x\_case <- as.matrix(temp\_case)

#### control data

 temp\_control <- read.table("x0.txt", header=T) ######### in a file "x0.txt", store the control data with n=190 rows and my.p+my.other = 40 columns

 x\_ctrl <- as.matrix(temp\_control)

#### weight matrix

 weight.mat <- read.table("weight.txt", header=T) #### in a file "weight.txt" store the weight matrix with my.p rows and my.d columns

 weight\_mat=as.matrix(weight.mat)

 data.input<-list("n"=n, "my.p"=my.p, "my.d"=my.d, "Y"=Y, "x\_case"=x\_case, "x\_ctrl" = x\_ctrl, "weight\_mat"=weight\_mat)

##########

# RIDGE

##########

 inits.input<- function(){

 list( d =c(0,0,0, 0), t\_d\_val=1, tau\_b\_val=1,

 beta = rep(0,my.p),

 beta\_other = rep(0,my.other) ) }

#### para.to.save specifies the set of parameters estimates to be saved by rjags

 para.to.save<-c("beta","t\_d\_val","tau\_b\_val")

##########

# LASSO

#

# uncomment the following comands to run bayesian lasso

##########

# inits.input<- function(){

# list( d =c(0,0,0,0), tg2= rep(1,my.p),t\_b=1, lambda.2=1,

# beta = rep(0, my.p),

# beta\_other = rep(0, my.other))}

# para.to.save<-c("beta", "tg2", "t\_b", "lambda.2")

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

#### RUN MCMC

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

 n.iter.input = 50000

 n.thin.input = 50

 n.burn.input = 4000

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

# BAYESIAN RIDGE REGRESSON

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

 my.jags.ridge <- jags.model(winbug.func,sep, data=data.input,

 n.chains=1, n.adapt=n.burn.input, inits=inits.input)

 my.coda <- coda.samples(my.jags.ridge, variable.names=para.to.save, n.iter=n.iter.input, thin=n.thin.input)

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

# BAYESIAN LASSO

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

# my.jags.model.gamma <- jags.model(winbug.func, data=data.input,

# n.chains=1, n.adapt=n.burn.input, inits=inits.input)

# my.coda.gamma <- coda.samples(my.jags.model.gamma, variable.names=para.to.save, n.iter=n.iter.input, thin=n.thin.input)