Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

RESEARCH ARTICLE

Modeling a Bivariate Residential-Workplace Neighborhood Effect when Estimating the Effect of Proximity to Fast-Food Establishments on Body Mass Index

A. James O'Malley*[1,2] | Peter James[3,4] | Todd A. MacKenzie[1,2] | Jinyoung Byun[1] | Subu V. Subramanian[3,4] | Jason P. Block[5,6]

[1]Department of Biomedical Data Science, Geisel School of Medicine at Dartmouth, New Hampshire, USA
[2]The Dartmouth Institute of Health Policy and Clinical Practice, Geisel School of Medicine at Dartmouth, New Hampshire, USA
[3]Department of Epidemiology and Environmental Health, Harvard TH Chan School of Public Health, Massachusetts, USA
[4]Department of Environmental Health, Harvard TH Chan School of Public Health, Massachusetts, USA
[5]Department of Population Medicine, Harvard Pilgrim Health Care Institute, Massachusetts, USA
[6]Department of Population Medicine, Harvard Medical School, Massachusetts, USA

Correspondence
*A. James O'Malley, The Dartmouth Institute of Health Policy and Clinical Practice, Geisel School of Medicine at Dartmouth, 1 Medical Center Drive, Geisel School of Medicine at Dartmouth, Lebanon, NH 03756, USA. Email: James.OMalley@Dartmouth.edu

Abstract

The Supplementary Materials available at this GitHub site contain details of the R and JAGS code for estimating the model. Instructions for running the code on the synthetic data set we have developed are also provided.

1 | STUDY DATA

Data from the Framingham Heart Study is not freely available due to patient confidentiality. The Framingham Heart Study has a formal process for requesting data, and anyone can request data directly via: https://www.framinghamheartstudy.org/researchers/application-review.php. In lieu of the actual data, we provide a pseudo data set named simdata.txt. The size and structure of these data are identical to the actual data, allowing the same code to be used to replicate the Bayesian analysis that yielded the results reported in the main text. The pseudo data, data description, and code

are available on GitHub at https://github.com/kiwijomalley/ProximityToFood. For readers' convenience, the data description and code are also provided below in PDF format.

1.1 | Data dictionary

The data is described in the following:

- ID: The identification number of the individual. Ranges from 1 to 2,889.
- wave: The Framingham Heart Study exam wave; ranges from 1 to 8 for the Offspring cohort and 6 to 8 for the Omni cohort, which only started in wave 6 of the offspring cohort
- Tobs: The total number of exams the individual attended
- neighborhoodHome: The neighborhood where the individual lived
- neighborhoodWork: The neighborhood where an individual was employed (if employed)
- BMI: Body Mass Index
- yob: Year of birth of individual
- smokes: Whether individual is a current smoker
- male: Whether individual has male gender
- married: Whether individual is currently married
- educ: Educational level of individual (0 = High-school or less, 1 = Completed high school; 2 = Other)
- tractpov: Percent of households below the poverty line in neighborhood where individual lived
- etractpov: Percent of households below the poverty line in neighborhood where individual was employed
- unemploy: Whether or not individual was unemployed
- DistHome: Distance in kilometers from individual's home to the nearest fast food establishment
- DistWork: Distance in kilometers from individual's workplace to the nearest fast food establishment (if employed)
- DriveDist: Number of fast food restaurants within a 60 meter buffer of the shortest commute between work and home

The first row of the data set contains the above variable names. Workplace distance, driving distance and the poverty of an individual's workplace neighborhood are only available for employed individuals. For simplicity, real numerical values are given for these variables on such cases. However, the JAGS script is coded so that only observations in which the individual is employed directly contribute to the estimation of model parameters involving these variables.

2 | CODE

The code for performing the analysis is shown below. The first subsection contains the R code in which the commands to call JAGS are specified, the call to JAGS, and finally post-estimation analysis and presentation of the results. The second subsection shows the JAGS code for the primary analysis. The entire analysis is performed by running the R code as JAGS is called from the R script.

2.1 | R Code

```
################################################################################################
# Runs analysis in R using RJAGS
# Must first install JAGS and then install RJAGS package from CRAN website
################################################################################################

## R wrapper code to estimate distance-to-food model ##
library(rjags)
coda.options(combine.plots=TRUE,combine.stats=TRUE)

#Specify filename of JAGS code for Bayesian analysis and data in and output directories
rsource="/Volumes/STORAGE/JOMalley/Dartmouth/Biostatistics/DistanceToFood/Code" #If run on a Linux server
setwd(rsource)
datdir="../Data/" #Directory to store data
outdir="../Output/" #Directory to store output

#Specify code and initial values to use
IW=1
subtype=1
if (IW==1) {
 if (subtype==1) {
  JAGScode<-"RealBMICodeIntIW.bug"
  #JAGScode<-"RealBMICodeIntIWlog.bug"
 } else if (subtype==2) {
  JAGScode<-"RealBMICodeIntArea0.bug"
 } else if (subtype==3) {
  JAGScode<-"RealBMICodeIntRSlope0.bug"
 } else {
  JAGScode<-"RealBMICodeIntSerial0.bug"
 }
 IC<-list(be=c(30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), isigma=0.25, u=0.5) #Initial values
} else {
 JAGScode<-"RealBMICodeIntProdNorm.bug"
 IC<-list(be=c(30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), isigma=0.25, ipntauarea=c(1, 0.1), ipntaurand=c(1, 0.1), psiran
}

#Read in data
datatype=1
if (datatype==1) {
 #BMIdata<-read.table(paste(datdir,"simdata.txt",sep=""),sep=" ",col.names=c("ID", "wave", "Omni", "Tobs", "TractHome", "TractWork", "BMI", "yob", "s
 BMIdata<-read.table(paste(datdir,"datawbugs4town.txt",sep=""),sep=" ",col.names=c("ID", "wave", "Omni", "Tobs", "TractHome", "TractWork", "BMI", "yo
} else if (datatype==2) {
 BMIdata<-read.table(paste(datdir,"datawbugs14town.txt",sep=""),sep=" ",col.names=c("ID", "wave", "Omni", "Tobs", "TractHome", "TractWork", "BMI", "y
} else {
 BMIdata<-read.table(paste(datdir,"datawbugs4towncf.txt",sep=""),sep=" ",col.names=c("ID", "wave", "Omni", "Tobs", "TractHome", "TractWork", "BMI", "
}

#Define parameters and operating characteristics of MCMC procedure
parameters = c("be", "sigma", "rho", "taurand", "tauarea", "corrand", "corarea", "Rsquare")    # The parameter(s) to be monitored.
adaptSteps = 500              # Number of steps to "tune" the samplers.
burnInSteps = 20000           # Number of steps to "burn-in" the samplers.
nChains = 3                   # Number of chains to run.
numSavedSteps=150000          # Total number of steps in chains to save.
thinSteps=1                   # Number of steps to "thin" (1=keep every step).
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains ) # Steps per chain.

#Make an object for fitting model using JAGS
jagsModel = model<-jags.model(JAGScode,data=BMIdata,inits=IC,n.chains=nChains)

# Burn-in phase of model estimation:
cat( "Burning in the MCMC chain...\n" )
update( jagsModel , n.iter=burnInSteps )

# Main phase of MCMC model: draws from the posterior distribution are saved in codaSamples
cat( "Sampling final MCMC chain...\n" )
codaSamples = coda.samples( jagsModel , variable.names=parameters ,
                            n.iter=nIter , thin=thinSteps )
# resulting codaSamples object has these indices:
#   codaSamples[[ chainIdx ]][ stepIdx , paramIdx ]

mcmcChain = as.matrix( codaSamples )
summary(mcmcChain)
```

```
## Prepare for and implement post model-fitting computations to check whether chain converged ##

beSample0 = mcmcChain[,"be[1]"] # Put sampled values in a vector.
beSample1 = mcmcChain[,"be[2]"] # Put sampled values in a vector.
sigmaSample = mcmcChain[,"sigma"] # Put sampled values in a vector.
rhoSample = mcmcChain[,"rho"] # Put sampled values in a vector.
tau1Sample = mcmcChain[,"tauarea[1,1]"] # Put sampled values in a vector.
corrandSample = mcmcChain[,"corrand"] # Put sampled values in a vector.
corareaSample = mcmcChain[,"corarea"] # Put sampled values in a vector.

#Plots of sequences of draws
par(mfrow=c(3,2), srt=0, mai=c(0.6, 0.6, 0.4, 0.2), mgp=c(2,1,0))
plot(beSample0[1:nIter],type="l",main="MCMC Chain for Intercept",xlab="iteration",ylab="estimate")
plot(beSample1[1:nIter],type="l",main="MCMC Chain for Wave 2",xlab="iteration",ylab="estimate")
plot(sigmaSample[1:nIter],type="l",main="MCMC Chain for error variance",xlab="iteration",ylab="estimate")
plot(tau1Sample[1:nIter],type="l",main="MCMC Chain for home variance",xlab="iteration",ylab="estimate")
plot(corrandSample[1:nIter],type="l",main="MCMC Chain for individual RE correlation",xlab="iteration",ylab="estimate")
plot(corareaSample[1:nIter],type="l",main="MCMC Chain for area RE correlation",xlab="iteration",ylab="estimate")
dev.copy2eps(file=paste(outdir,"TraceSeqIW4.eps",sep=""), width=6, height=6, horizontal=FALSE) #to file

#Trace plots of long-run averages
par(mfrow=c(1,1), srt=0, mai=c(0.6, 0.6, 0.4, 0.2), mgp=c(2,1,0))
trpSample=cbind(corareaSample[1:nIter],corareaSample[(nIter+1):(2*nIter)],corareaSample[(2*nIter+1):(3*nIter)])
for (i in 2:nIter) {
 trpSample[i,1]=((i-1)*trpSample[i-1]+corareaSample[i])/i
 trpSample[i,2]=((i-1)*trpSample[nIter+i-1]+corareaSample[nIter+i])/i
 trpSample[i,3]=((i-1)*trpSample[2*nIter+i-1]+corareaSample[2*nIter+i])/i
}
mn=mean(corareaSample); sd=sqrt(var(corareaSample))
plot(trpSample[,1],type='l',ylim=c(mn-0.25*sd,mn+0.25*sd),xlim=c(0,nIter),
 main='Trace plot of posterior mean for 3 chains',
 xlab='Draw',
 ylab='Mean')
lines(trpSample[,2],type='l',col='red')
lines(trpSample[,3],type='l',col='green')
dev.copy2eps(file=paste(outdir,"TracePlotIW4.eps",sep=""), width=6, height=6, horizontal=FALSE) #to file

#Autocorrelation plots
par(mfrow=c(1,1), srt=0, mai=c(0.6, 0.6, 0.4, 0.2), mgp=c(2,1,0))
autocorr.plot(corareaSample[1:nIter],main="MCMC Chain for p",auto.layout=FALSE)
dev.copy2eps(file=paste(outdir,"AutoCorrPlotIW4.eps",sep=""), width=6, height=6, horizontal=FALSE) #to file

#Gelman-Rubin MCMC convergence statistic
MCMCdiag=gelman.diag(codaSamples,confidence=0.95,autoburnin=FALSE,multivariate=FALSE)$psrf

#Residual analysis
rsquare=mcmcChain[,"Rsquare"]
print(summary(rsquare))

## Generate statistical inferences ##

#Plots of drawn values
par(mfrow=c(3,2), srt=0, mai=c(0.6, 0.6, 0.4, 0.2), mgp=c(2,1,0))
plot(density(mcmcChain[,"be[18]"]),xlab="",ylab="Density",
 main="Home distance to FF effect for females")
plot(density(mcmcChain[,"be[21]"]),xlab="",ylab="Density",
 main="Work distance to FF effect for males")
plot(density(mcmcChain[,"tauarea[1,1]"]),xlab="",ylab="Density",
 main="Residential Variance")
plot(density(mcmcChain[,"tauarea[2,2]"]),xlab="",ylab="Density",
 main="Workplace Variance")
plot(density(mcmcChain[,"corrand"]),xlab="",ylab="Density",
 main="Correlation of individual random effects")
plot(density(mcmcChain[,"corarea"]),xlab="",ylab="Density",
 main="Correlation of area random effects")
dev.copy2eps(file=paste(outdir,"FixedSerialIW4.eps",sep=""), width=6, height=6, horizontal=FALSE) #to file

#Summary statistics of draws (the types of things you might put in a paper)
mn=apply(mcmcChain,2,mean)
stdev=sqrt(apply(mcmcChain,2,var))
lowl=apply(mcmcChain,2,quantile,0.025)
uppl=apply(mcmcChain,2,quantile,0.975)
pg0=apply(mcmcChain,2,function(x) mean(x>0))
```

```
sumdata=cbind(mn,stdev,lowl,uppl,pg0)
write.table(t(sumdata),file=paste(outdir,"BayesSummaryStatisticsIW4.txt",sep=""))
```

2.2 | JAGS code

```
model {
   BMI[1] ~ dnorm(la[1],isigma.e[1]);
   la[1] <- pred[1];
   pred[1] <- be[1] + waveterms[1] + be[9]*male[1] + be[10]*cyob[1] + be[11]*smokes[1] + be[12]*married[1] + be[13]*educ1[1]
       + be[14]*educ2[1] + be[15]*employ[1] + be[16]*ctractpov[1] + be[17]*cetractpov[1]*employ[1]
       + (be[18]*female[1] + be[19]*male[1])*cDistHome[1] + (be[20]*female[1] + be[21]*male[1])*cDistWork[1]
       + (be[22]*female[1] + be[23]*male[1])*cDriveDist[1]
       + mu[ID[1], 1] + mu[ID[1], 2]*wave[1] + th[TractHome[1], 1] + th[TractWork[1], 2]*employ[1];
   isigma.e[1] <- isigma;

   waveterms[1] <- be[2]*wave2[1] + be[3]*wave3[1] + be[4]*wave4[1] + be[5]*wave5[1]
                  + be[6]*wave6[1] + be[7]*wave7[1] + be[8]*wave8[1];
   wave2[1] <- equals(wave[1], 2);
   wave3[1] <- equals(wave[1], 3);
   wave4[1] <- equals(wave[1], 4);
   wave5[1] <- equals(wave[1], 5);
   wave6[1] <- equals(wave[1], 6);
   wave7[1] <- equals(wave[1], 7);
   wave8[1] <- equals(wave[1], 8);
   educ1[1] <- equals(educ[1], 1);
   educ2[1] <- equals(educ[1], 2);
   female[1] <- 1 - male[1];
   employ[1] <- 1 - unemploy[1];
   cyob[1] <- yob[1] - mean(yob[]);
   ctractpov[1] <- tractpov[1] - mean(tractpov[]);
   cetractpov[1] <- (etractpov[1] - mean(etractpov[]))*employ[1];
   cDistHome[1] <- DistHome[1] - mean(DistHome[]);
   cDistWork[1] <- (DistWork[1] - mean(DistWork[]))*employ[1]; #Only non-zero for employed individuals
   cDriveDist[1] <- (DriveDist[1] - mean(DriveDist[]))*employ[1]; #Only non-zero for employed individuals
   Omni[1] ~ dunif(0,1);
   Tobs[1] ~ dunif(0,8);
   R2[1] <- pow(BMI[1] - pred[1], 2);

   for (i in 2:length(ID)) {
      BMI[i] ~ dnorm(la[i],isigma.e[i]);
      la[i] <- pred[i] + rho*(BMI[i-1] - pred[i-1])*nfst[i];
      pred[i] <- be[1] + waveterms[i] + be[9]*male[i] + be[10]*cyob[i] + be[11]*smokes[i] + be[12]*married[i] + be[13]*educ1[i]
          + be[14]*educ2[i] + be[15]*employ[i] + be[16]*ctractpov[i] + be[17]*cetractpov[i]*employ[i]
          + (be[18]*female[i] + be[19]*male[i])*cDistHome[i] + (be[20]*female[i] + be[21]*male[i])*cDistWork[i]
          + (be[22]*female[i] + be[23]*male[i])*cDriveDist[i]
          + mu[ID[i], 1] + mu[ID[i], 2]*wave[i] + th[TractHome[i], 1] + th[TractWork[i], 2]*employ[i];
      isigma.e[i] <- isigma/(1 - pow(rho, 2)*nfst[i]);
      nfst[i] <- 1 - step(ID[i] - ID[i-1] - 0.5); #Subtract 0.5 for the exact inequality

      waveterms[i] <- be[2]*wave2[i] + be[3]*wave3[i] + be[4]*wave4[i] + be[5]*wave5[i]
                     + be[6]*wave6[i] + be[7]*wave7[i] + be[8]*wave8[i];
      wave2[i] <- equals(wave[i], 2);
      wave3[i] <- equals(wave[i], 3);
      wave4[i] <- equals(wave[i], 4);
      wave5[i] <- equals(wave[i], 5);
      wave6[i] <- equals(wave[i], 6);
      wave7[i] <- equals(wave[i], 7);
      wave8[i] <- equals(wave[i], 8);
      educ1[i] <- equals(educ[i], 1);
      educ2[i] <- equals(educ[i], 2);
      female[i] <- 1 - male[i];
      employ[i] <- 1 - unemploy[i];
      cyob[i] <- yob[i] - mean(yob[]);
      ctractpov[i] <- tractpov[i] - mean(tractpov[]);
      cetractpov[i] <- (etractpov[i] - mean(etractpov[]))*employ[i];
      cDistHome[i] <- DistHome[i] - mean(DistHome[]);
      cDistWork[i] <- (DistWork[i] - mean(DistWork[]))*employ[i]; #Only non-zero for employed individuals
      cDriveDist[i] <- (DriveDist[i] - mean(DriveDist[]))*employ[i]; #Only non-zero for employed individuals
      Omni[i] ~ dunif(0,1);
      Tobs[i] ~ dunif(0,8);
      R2[i] <- pow(BMI[i] - pred[i], 2);
   }
```

```
   #Distribution for random effects
   for (j in 1:max(ID)) {
      mu[j,1:2] ~ dmnorm(mn1[], itaurand[,]);
   }

   #Distribution for area effects
   for (k in 1:max(TractHome)) {
      th[k,1:2] ~ dmnorm(mn2[],itauarea[,]);
   }

   #Prior for fixed effects
   for (k in 1:23) {
      be[k] ~ dnorm(0,1.0E-6);
   }

   #Hyper-priors
   isigma ~ dgamma(1.0E-3,1.0E-3);
   itaurand[1:2,1:2] ~ dwish(Omrand[,],df);
   itauarea[1:2,1:2] ~ dwish(Omarea[,],df); #Make 2nd param = df
   df <- 4 #2, 4 or 13 for second parameter
   u ~ dbeta(1,1); #dbeta(1,1) = unif
   rho <- 2*u - 1;

   mn1[1] <- 0; mn1[2] <- 0;
   mn2[1] <- 0; mn2[2] <- 0;
   Omrand[1,1] <- 1; Omrand[1,2] <- 0; Omrand[2,1] <- 0; Omrand[2,2] <- 1;
   Omarea[1,1] <- OmareaMn; Omarea[1,2] <- 0; Omarea[2,1] <- 0; Omarea[2,2] <- OmareaMn;
   OmareaMn <- 1*(df - 3); #To ensure prior mean is 0.1, supply inverse of scaled mean to Wishart

   #Inverse of variances and covariance matrices
   sigma <- 1.0/isigma;
   taurand[1:2,1:2] <- inverse(itaurand[,]);
   tauarea[1:2,1:2] <- inverse(itauarea[,]);
   corrand <- taurand[1,2]/sqrt(taurand[1,1]*taurand[2,2]);
   corarea <- tauarea[1,2]/sqrt(tauarea[1,1]*tauarea[2,2]);

   #Derived quantities of interest
   iccind <- taurand[1,1]/(taurand[1,1]+sigma);
   bothfdist <- (1 - step(be[18]))*(1 - step(be[21])); #Female home and male work
   onefdist <- 1 - step(be[18])*step(be[21]);
   Rsquare <- 1-mean(R2[])/pow(sd(BMI[]),2);
}
```