## Supplementary Text 1. Code for Sales Prediction Model

# Code were run by OpenBUGS software

#Description and example code for proper Conditional Autoregressive (CAR) Model are also available in GeoBugs manual. https://www.mrc-bsu.cam.ac.uk/wp-content/uploads/geobugs12manual.pdf

#OpenBUGS code to generate sales of plain yogurt (same code can be applied to the other food categories)

# YOGPL stands for plain yogurt.

# Note that Y\_YOGPL correspond to Q\_ij in the main text and contain observed and missing sales

model{

### Non-sugar yogurt sales #########################################################################

for(j in 1:J\_YOGPL){ #J\_YOGPL inicdates number of stores

# log sales of soda as normal

logY\_YOGPL[j] ~ dnorm(mu.y\_YOGPL[j], tau.y\_YOGPL)

# Mean of log sales specified by the sum of chain random effect and area-level effect for area

# ID\_chain\_YOGPL is an index for chain where store j belongs to

# ID\_neigh\_YOGPL is an index for area where store j belongs to

mu.y\_YOGPL[j] <- betaChain\_YOGPL[ID\_chain\_YOGPL[j]] + Z\_YOGPL[ID\_neigh\_YOGPL[j]]

# generate non-log soda sales

Y\_YOGPL[j] <- exp(logY\_YOGPL[j])

}

# Area-level effects Z containing area-level covariates and spatial random effect S\_YOGPL

# N\_neigh is number of areas (n=193)

for(i in 1:N\_neigh) {

Z\_YOGPL[i] <-

betaEduc\_YOGPL\*educ\_neigh[i] + betaIncome\_YOGPL\*income\_neigh[i]+ betaPopDensity\_YOGPL\*popDensity\_neigh[i] +

betaFamSize\_YOGPL\*famSize\_neigh[i] + betaYoung\_YOGPL\*young\_neigh[i] + betaEmpRate\_YOGPL\*empRate\_neigh[i] + S\_YOGPL[i]

}

# Specification of proper CAR prior for spatially structured area-level random effect. See the link above for details

S\_YOGPL[1:N\_neigh] ~ car.proper(mu[], C[], adj\_neigh[], num\_neigh[], M[], tau.car\_YOGPL, gamma)

for(i in 1:N\_neigh){

mu[i]<-betaZeroArea\_YOGPL

M[i] <- 1/num\_neigh[i]

}

cumsum[1] <- 0

for(i in 2:(N\_neigh+1)) { cumsum[i] <- sum(num\_neigh[1:(i-1)]) }

for(k in 1 : sumNumNeigh) {

for(i in 1:N\_neigh) { pick[k,i] <- step(k - cumsum[i] - epsilon) \* step(cumsum[i+1] - k)

}

C[k] <- 1 / inprod(num\_neigh[], pick[k,])

}

epsilon <- 0.0001

gamma.min <- min.bound(C[], adj\_neigh[], num\_neigh[], M[])

gamma.max <- max.bound(C[], adj\_neigh[], num\_neigh[], M[])

gamma ~ dunif(gamma.min, gamma.max)

# Store chain random effect

for(k in 1:J\_YOGPL\_chain){

betaChain\_YOGPL[k] ~dnorm(betaZero\_YOGPL, tau.chain\_YOGPL)

}

# Standard deviations

sigma.chain\_YOGPL <-1/sqrt(tau.chain\_YOGPL)

sigma.car\_YOGPL <- sqrt(1/tau.car\_YOGPL)

sigma.y\_YOGPL <- sqrt(1/tau.y\_YOGPL)

#Prior probabilities

tau.y\_YOGPL ~ dgamma(0.5, 0.01)

tau.car\_YOGPL ~ dgamma(0.5, 0.01)

tau.chain\_YOGPL ~ dgamma(0.5, 0.01)

betaZero\_YOGPL ~ dnorm(0,0.01)

betaZeroArea\_YOGPL <- 0

betaEduc\_YOGPL ~ dnorm(0,0.01)

betaIncome\_YOGPL ~ dnorm(0,0.01)

betaPopDensity\_YOGPL ~ dnorm(0,0.01)

betaFamSize\_YOGPL ~ dnorm(0,0.01)

betaYoung\_YOGPL ~ dnorm(0,0.01)

betaEmpRate\_YOGPL ~ dnorm(0,0.01)

# End of Sales prediction model for yogurt

### Extra step- indicator generation

# Multiply the column vector of yogurt (plain) sales with matrix whose elements are a pairwise area-level store visit probability, between store and area multiplied with population identity (matrix G\* in the main text)

# N\_Neigh is the number of area (neighborhoods),, 193

# n\_Neigh is an iterator (index) for area i

# odMX\_ YOGPL is matrix G\*

# Y\_ YOGPL is exponentiated sales quantity of stores selling yogurt

for(n\_Neigh in 1:N\_Neigh){

#Get linear combination of all area’s sales multiplied with the OD weights for eatch area N\_Neigh

# Note that OpenBugs does not allow matrix multiplication, thus a series of inprod() – inner products - is needed.

neighSales\_combined\_ YOGPL[n\_Neigh] <- inprod(Y\_YOGPL[1:J\_OD\_YOGPL], odMX\_ YOGPL[,n\_Neigh])

# finally divide the value (purchased quantity in an area n\_neigh by population in each area to make the quantity purchased per resident

# indicator\_ YOGPL corresponds to X in the main text

indicator\_ YOGPL[n\_Neigh] <- neighSales\_combined\_ YOGPL[n\_Neigh]/ pop\_Neigh[n\_Neigh]

}

#End of script

}