|  |
| --- |
| **Spatial data analysis (in R)**  **prof.ucz.dr hab. Katarzyna Kopczewska**  **Class 02 –** spatial weights matrix  building neighborhood relations |

1. **Basic types of neighborhood (NB) matrices and the spatial weights (W) matrices**

A key element of spatial modeling is the **structure of the neighborhood**, on which the modeling of the neighbor interaction is based. In practice, one uses few types of neighborhood matrices in modelling:

- Contiguity matrix (matrix of common border)

- Matrix of k nearest neighbors

- Matrix of neighbors within d km (neighbors in radius of d km)

Neighborhood matrices define a neighborhood with dummies. NB matrices do not take into account the weights - strength of spatial interaction.

One adds the weights to obtain the spatial weights matrix. Weighing of neighborhood relations is most often performed in two ways:

- row-standardizing to 1 (ie. dividing the values in the row by the number of elements in a row)

- weighing with the distance (inverse distance)

On this basis, there are following basic types of spatial weight matrices:

- Contiguity matrix (row standardized to one)

- Matrix of k nearest neighbors (knn)

- Matrix of neighbors within d km (d near neigh…)

- Inverse distance matrix (*de facto* knn for k = n-1 weighted by distance)

- Inverse squared distance matrix (*de facto* KNN for k = n-1 weighted square of the distance)

The choice of the spatial weights matrix modeling is an important step. When a-priori selected matrix does not correspond to a real relationship, there is a need to find more adequate spatial weights matrix.

1. **Basic functions in R for handling the spatial weights matrix**

poly2nb Construct neighbours list from polygon list

nb2listw Spatial weights for neighbours lists

knearneigh K nearest neighbours for spatial weights

dnearneigh Neighbourhood contiguity by distance

knn2nb Neighbours list from knn object

is.symmetric.nb Test a neighbours list for symmetry

make.sym.nb(nb) Makes a non-symmetric list symmetric by adding neighbors

lag.listw Spatial lag of a numeric vector

nbdists Spatial link distance measures

nb2mat Spatial weights matrices for neighbours lists

mat2listw Convert a square spatial weights matrix to a

nblag Higher order neighbours lists

**General rule:**

**-** for drawing use objects of class **nb**

**-** for calculations use objects of class **listw**

1. **Sample codes in R – spatial weights matrix**

**# Spatial weights matrix – contiguity matrix**

cont.nb<-poly2nb(as(pov, "SpatialPolygons"))

cont.listw<-nb2listw(cont.nb, style="W")



cont.listw # summary of matrix

**# coordinates of nts4 units**

**#(geometric center of gravity)**

crds<-coordinates(pov)

colnames(crds)<-c("cx", "cy")

**# plot of neighbourhood**

plot(pov) # contour map

plot(cont.nb, crds, add=TRUE)

# conversion to class matrix

cont.mat<-nb2mat(cont.nb)

cont.mat[1:10, 1:10]

**# spatial weights matrix – k nearest neighbours**

nts4.knn<-knearneigh(crds, k=1) # knn class (k=1)

nts4.knn.nb<-knn2nb(nts4.knn)



plot(nts4)

plot(nts4.knn.nb, crds, add=TRUE)

**# checking for matrix symmetry**

print(is.symmetric.nb(nts4.knn.nb))

nts4.sym.knn.nb<-make.sym.nb(nts4.knn.nb)

print(is.symmetric.nb(nts4.sym.knn.nb))

**# knn as listw class**

nts4.sym.knn.listw<-nb2listw(nts4.sym.knn.nb)

**# spatial weights matrix – neighbours in radius of d km**

conti30<-dnearneigh(crds, 0, 30,longlat=TRUE) # conti30 is nb class

plot(pov)

plot(conti30, crds, add=TRUE)

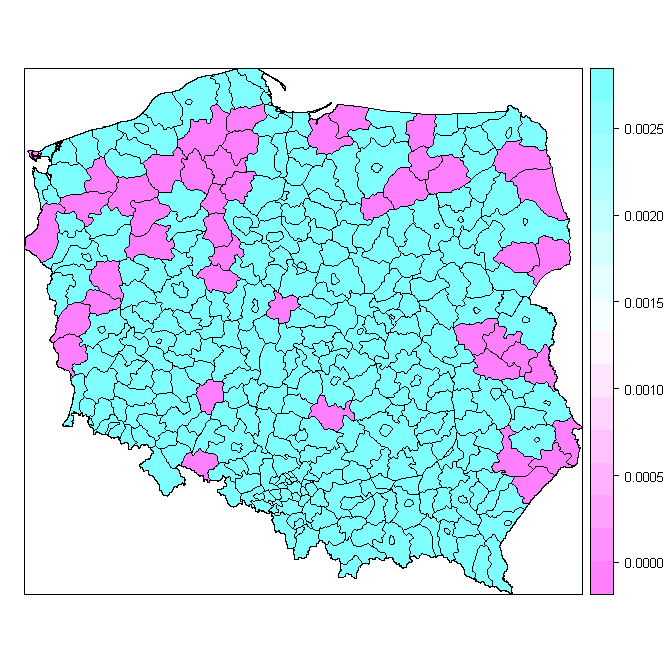
conti30.m<-nb2mat(conti30, zero.policy=TRUE)

**# let’s see who does not have a neighbour**

a<-colMeans(t(conti30.m))

nts4$a<-a

spplot(nts4, "a")



**# at what distance all units have at least one neighbour?**

# implicitly k=1, list of closests neighbours

kkk<-knn2nb(knearneigh(crds))

# max of distances between clostests neighbours 🡪 we get distance

all<-max(unlist(nbdists(kkk, crds)))

# neighbours in radius of d km

# guaranted that all regions have at least one neighbor

all.nb<-dnearneigh(crds, 0, all)

summary(all.nb, crds)

plot(pov, border="grey")

plot(all.nb, crds, add=TRUE)

conti.all<-nb2listw(all.nb)

**# spatial weights matrix - inverse distance matrix**

nts4.knn<-knearneigh(crds, k=379) # we have 380 units on the map

nts4.nb<-knn2nb(nts4.knn)

dist<-nbdists(nts4.nb, crds)

dist1<-lapply(dist, function(x) 1/x) # object of listw class

nts4.dist.listw<-nb2listw(nts4.nb, glist=dist1) # listw class object – spatial weights according to distance criterion

# converting to matrix class

dist.mat<-listw2mat(nts4.dist.listw)

summary(dist.mat) # statistics of weights of all regions

**# plot of inverse distance weights for single region**

pov.df<-as.data.frame(pov)

pov.df$jpt\_nazwa # Warszawa is 151

x<-dist.mat[,151]

summary(x)

cols<-rev(heat.colors(11))

#brks<-(0:7)\*5

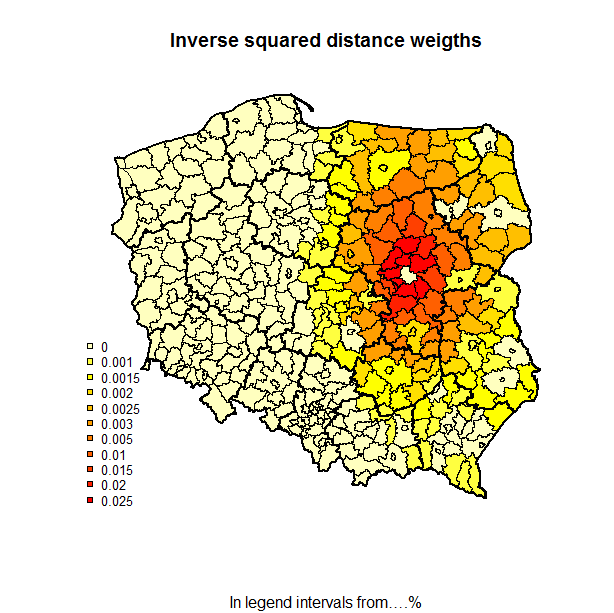
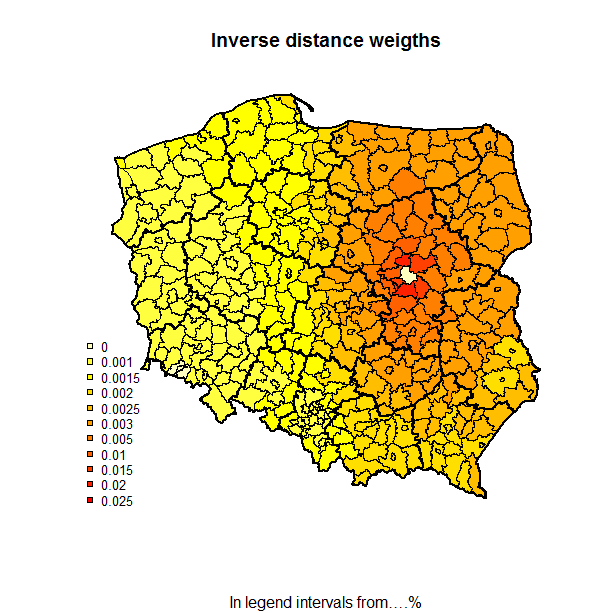
brks<-c(0, 0.001, 0.0015, 0.002, 0.0025, 0.003, 0.005, 0.01, 0.015, 0.02, 0.025)

plot(pov, col=cols[findInterval(x, brks)])

plot(voi, add=TRUE, lwd=2)

legend("bottomleft", legend=brks, pt.bg=cols, bty="n", pch=22, cex=0.8)

title(main="Inverse distance weigths", sub="In legend intervals from….%")



**# summary of spatial weights matrix**

summary(cont.nb) # matrix in nb class

cont.nb.listw<-nb2listw(cont.nb)

summary(unlist(cont.nb.listw$weights))

cont.nb.mat<-nb2mat(cont.nb)

summary(cont.nb.mat)

table(card(cont.nb)) # cardinalities, listing possible number of links and counting observations for given level

print(cont.nb.mat)

cont.listw$weights # lists spatial weights

**Task 1:** Visualise distance of all poviats from Suwalki

**Task 2:** Check how many neighbors have the counties (nts4 units, powiats) under criterion of neighbors within a radius of 70 km. Check out what changes were introduced in the matrix after make.symm.nb() command.

**Task 3:** For the contiguity matrix, draw a map with colors for the average weight of a neighbour