Guide for computing shortest paths with given points and base network map:

1. Download a package “shp2graph”

https://sourceforge.net/projects/shp2graph/

which is based on maptools , graph, igraph (I am afraid that you have to install them manually if you don’t have them, because this package is not connected with R Cran)

2. Assume you have data points “testpts.shp” and network data “testrn.shp” as shown in the figure 1

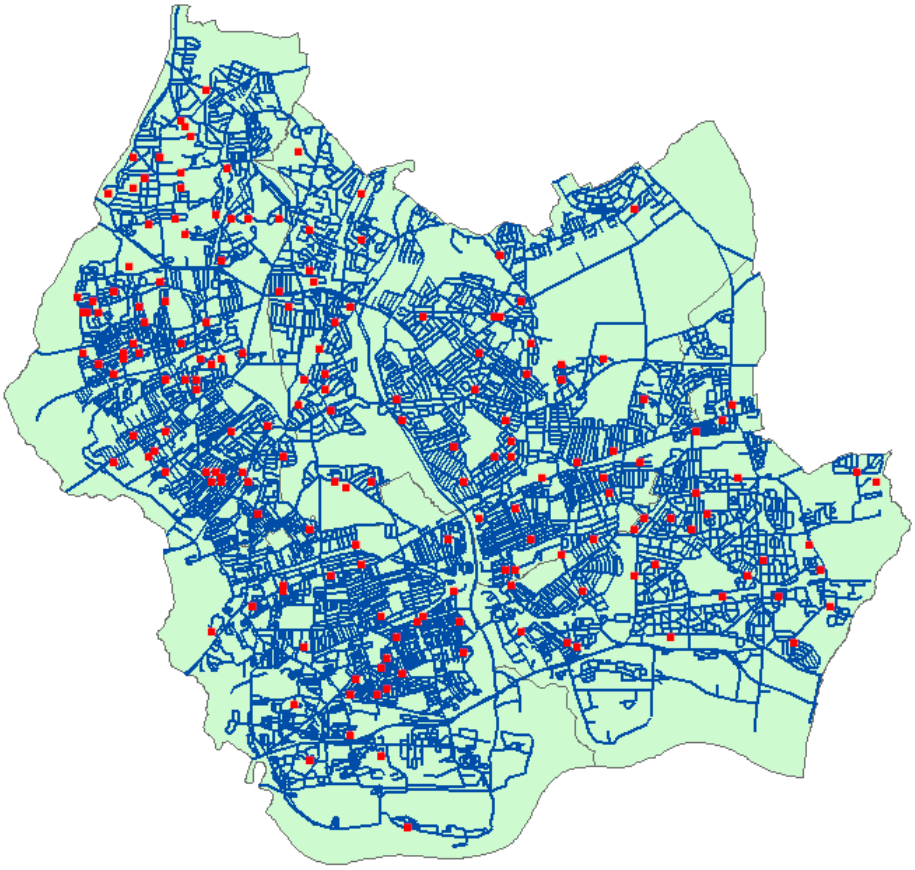


Figure 1: Example data points and network data

Read them in R

rn<-readShapeLines("testpts.shp", proj4string=CRS(as.character(NA)))

pts<-readShapePoints("testpts.shp”, proj4string=CRS(as.character(NA)))

3. Check the connectivity of the network data, because it is important for the final results.

If you are sure there is no topology error in the network data, skip this step

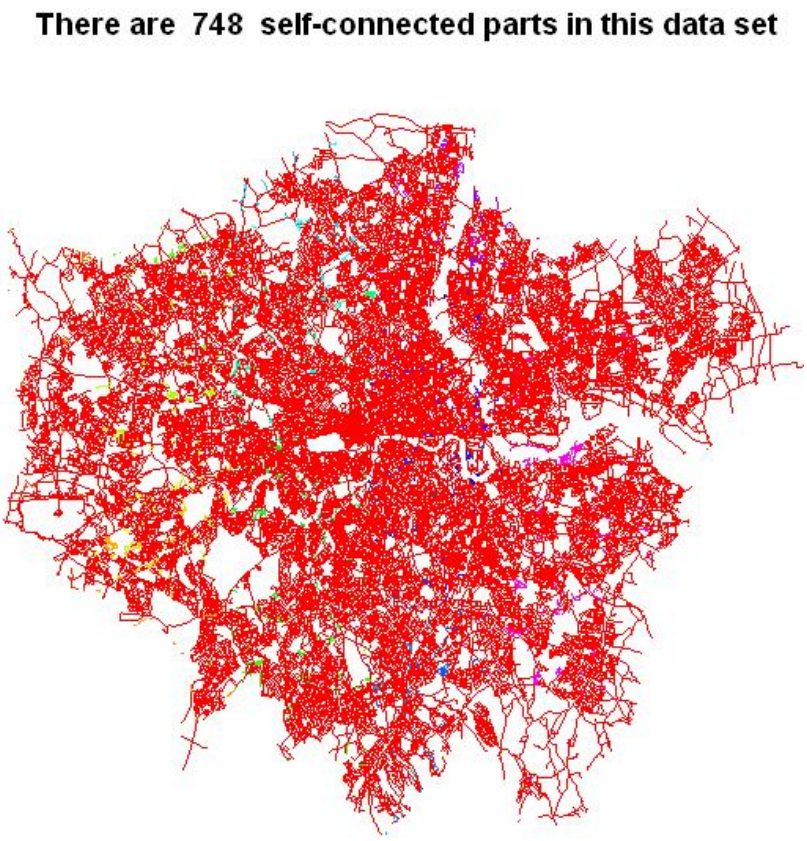
res<-nt.connect(rn) ## Check if the network is connected

Note: In this case the example network is connected, as figure 2 shows.

Figure 2 Connectivity of example network

However, the given network is not often self-connected, here are two bad examples:

Bad examples 1:



In this network, only small isolated parts are not connected to the main part, and you could ignore them. In this case, as the function “nt.connect” will return the main connected part of the road network as a “SpatialLinesDataFrame”, you could just use it in the next steps. Do

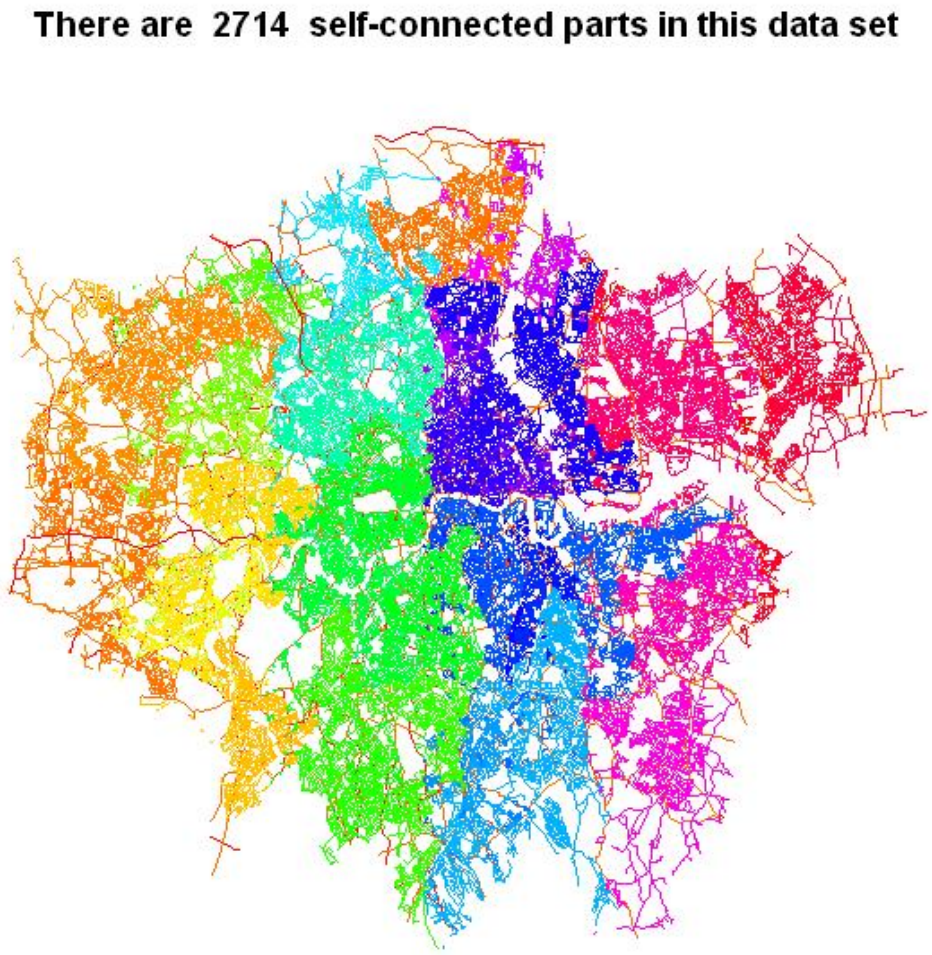
res<-nt.connect(rn)

rn<-res

##Or you can also save it as a new shape file

writeLinesShape(res, fname=”NewNetwork”)

Bad examples 2:



In this case, the road network are separated in many different parts, and you can't just use the main connected part. What you should do is to use some topology correction tools to deal with it, like GRASS or ET tools for ArcGIS.

After correction, come back to this step until you get a satisfactory connected network data set.

4. In many instances, involved points are not always nodes of a network or even they are not on a network, so you should find the corresponding nodes on the network. DO

pxys<- coordinates(pts)

#Mapping each point to the nearest node in the network/graph

pt2nt<-points2network(ntdata=rn,pointsxy=pxys, mapping.method=1, ELComputed=T)

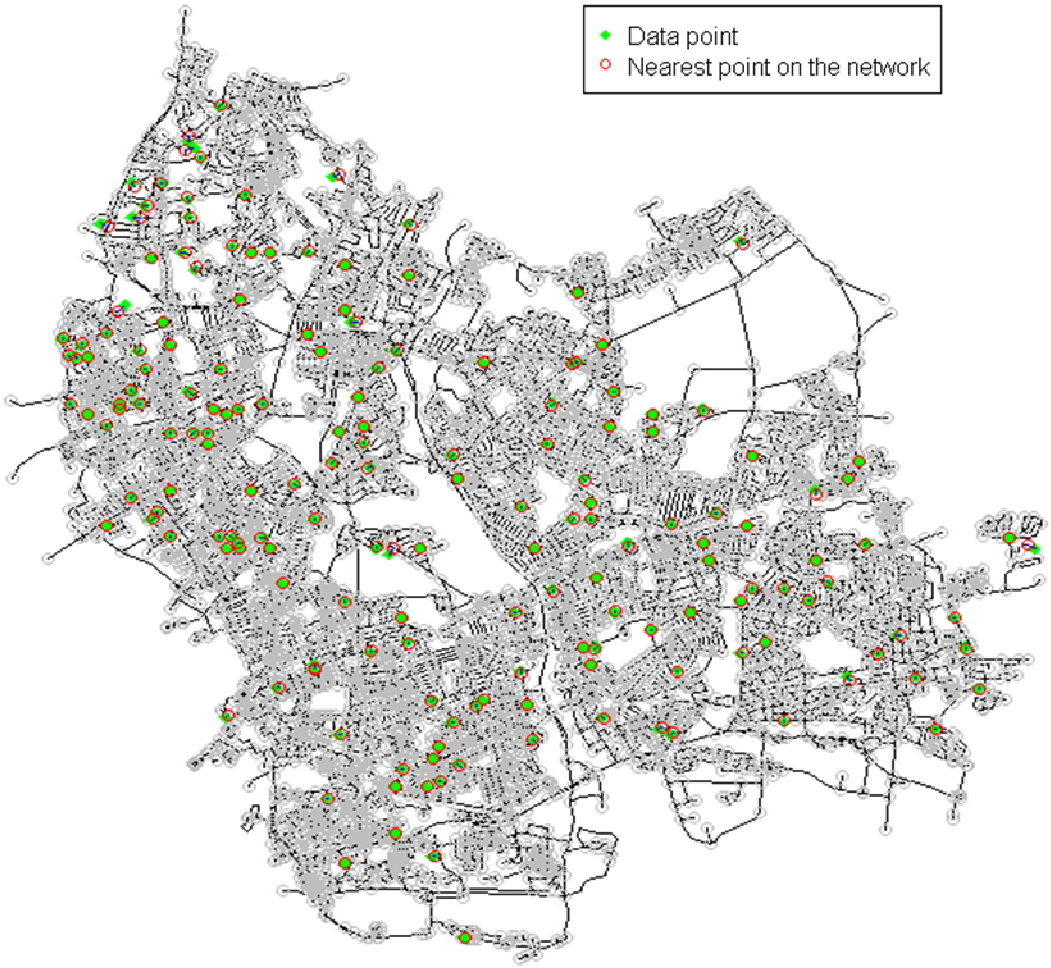
ptsinnt.view(ntdata=rn, nodelist= pt2nt [[1]], pointsxy=ptsxy, CoorespondIDs= pt2nt [[3]])

Or Mapping each point to the nearest point, if it is not a node of the network, it will be added as an node, which means the corresponding edge will be split into two parts

pt2nt<-points2network(ntdata=rn,pointsxy=pxys, mapping.method=2, ELComputed=T, ea.prop=rep(0,length(names(rn))))

ptsinnt.view(ntdata=rn, nodelist= pt2nt [[1]], pointsxy=ptsxy, CoorespondIDs= pt2nt [[3]])

In this example, you will see



5. Conversion from SpatialLines to a “graph “ or “igraph” object

CoorespondIDs= pt2nt [[3]] ##The IDs of corresponding nodes as for the data points.

###methods based on the igraph

###Generate an “igraph” object by weighting this graph with edge length

ig<-nel2igraph(pt2nt[[1]], pt2nt [[2]], weight= pt2nt [[8]])

###Compute the distance matrix from shortest paths with function “shortest.paths” in “igraph”

N<-length(CoorespondIDs)

dMat<-matrix(numeric(N*N),ncol=N)

Idxs<- as.numeric(CoorespondIDs) -1

for (i in 1:N)

{

spi<-shortest.paths(ig, v=c(Idxs[i]),weight=get.edge.attribute(ig,name="weight"))

dMat[i,]<-spi[as.numeric(CoorespondIDs)]

}

##

methods based on the “graph” and “RBGL”

###Generate an “graphNEL” object by weighting this graph with edge length

grNEL<-nel2graphNEL(pt2nt[[1]], pt2nt [[2]], weight= pt2nt [[8]])

library(RBGL)

N<-length(CoorespondIDs)

dMat<-matrix(numeric(N*N),ncol=N)

###Compute the distance matrix from shortest paths with function “sp.between” in “RBGL”

for (i in 1:(N-1))

{

for (j in (i+1):N)

{

dMat.nd[i,j]<-(sp.between(g2,start=as.character(CoorespondIDs[i]),finish=as.character(CoorespondIDs[j]) detail=F))[[1]]$length

dMat.nd[j,i]<-dMat.nd[i,j]

}

}

If any query, please email: lubinbin220@gmail.com