```
install.packages("maptools")
install.packages("spdep")
install.packages("rgdal")

setwd("E:/COA616_2018/Class5_09202018_SA_Rcode/") #the directory where you store
sids2.shp file and Boston.csv
library(maptools)
getinfo.shape("sids2.shp")

#Import shapefile
sids <- readShapePoly("sids2.shp")
class(sids)

#Assign a coordinate system (define)
proj4string(sids)<-CRS("+proj=longlat +ellps=WGS84")

#Transform to NAD83(HARN)
sids_NAD <- spTransform(sids,CRS("+init=epsg:3358"))
#http://spatialreference.org/ref/epsg/

#Transform to state plane NC (feet)
sids_SP <- spTransform(sids,CRS("+init=ESRI:102719"))
#http://spatialreference.org/ref/esri/

#www.spatialreference.org/ref/ for a list of CRS codes (stick with epsg and esri
codes)

#Create neighborhood

library(spdep)
sids_nbq <- poly2nb(sids) #Queen
sids_nbr <- poly2nb(sids, queen=FALSE) #Rook
coords <- coordinates(sids)

plot.new()
par(mfrow=c(2,1))  #divide plotting area into two rows and one column
plot(sids)
plot(sids_nbq,coords,add=T)

plot(sids)
plot(sids_nbr,coords,add=T)

#Point data
bost <- read.csv("boston.csv",header=T)
b.coord <- SpatialPoints(bost[,c("LON","LAT")])
bost2 <- SpatialPointsDataFrame(b.coord,bost)
coord_b <- coordinates(bost2)
bost_k <- knn2nb(knearneigh(coord_b,k=2,longlat=T)) #use k-nearest neighbors
```

```
algorithm

dev.off()
plot.new()
plot(as(bost2,"Spatial"),axes=T)
plot(bost_k,coord_b,add=T)
plot(bost2[bost2$CHAS==1,],col="blue",add=TRUE) #Select the points that border
Charles River

#Assign weights to the area that are linked - Creating spatial weights matrices
using neighborhood lists
#Row-standardized weights: row-standardization is used to create proportion
#weights in cases where features have an unequal number of neighbors.
#Divide each neighbor weight for a feature by the sum of all neighbor
#weights. Obs i has 3 neighbors, each has a weight of 1/3.
#Use it when you want comparable spatial parameters across different data
#sets with different connectivity structures.

sids_nbq_w <- nb2listw(sids_nbq,zero.policy=T) #default row-standardized weight
sids_nbq_w

#Binary weights
sids_nbq_wb <- nb2listw(sids_nbq,style="B")
sids_nbq_wb

# Weights based on IDW
dist <- nbdists(sids_nbq,coordinates(sids_SP))
idw <- lapply(dist,function(x) 1/(x/1000))
sids_nbq_idwb <- nb2listw(sids_nbq,glist=idw,style="B")
summary(unlist(sids_nbq_idwb$weights))

#Spatial autocorrelation
#Moran's I
moran.test(sids_NAD$SIDR79,listw=sids_nbq_w,alternative="two.sided")
moran.test(sids_NAD$SIDR79,listw=sids_nbq_wb)

#If you doubt that the assumptions of Moran's I are true (normality and
#randomization), we can use a Monte Carlo simulation
set.seed(1234)
bperm <- moran.mc(sids_NAD$SIDR79,listw=sids_nbq_w,nsim=999)
bperm
mean(bperm$res[1:999])  #expected Moran's I with random spatial pattern
var(bperm$res[1:999])
summary(bperm$res[1:999])
hist(bperm$res,freq=TRUE,breaks=20,xlab="Simulated Moran's I")
abline(v=0,col="red")
abline(v=bperm$res[1000],col="blue")
```

```
#Geary's C
geary.test(sids_NAD$SIDR79,listw=sids_nbq_w)
geary.test(sids_NAD$SIDR79,listw=sids_nbq_wb)
geary.test(sids_NAD$SIDR79,listw=sids_nbq_idwb)
```