Diagnosing spatial autocorrelation

Generate the data

First, input our weights matrix wm and our observations y.

```
wm <- cbind(c(0,1,0,1,1,0,0,0,0),
          c(1,0,1,1,1,1,0,0,0),
          c(0,1,0,0,1,1,0,0,0),
          c(1,1,0,0,1,0,1,1,0),
          c(1,1,1,1,0,1,1,1,1),
          c(0,1,1,0,1,0,0,1,1),
          c(0,0,0,1,1,0,0,1,0),
          c(0,0,0,1,1,1,1,0,1),
          c(0,0,0,0,1,1,0,1,0))
y <- c(90, 80, 60, 100, 100, 60, 100, 70, 80)
```

Calculate Moran's I

```
n <- length(y)
ybar <- mean(y)

# (yi-ybar)(yj-ybar) for all pairs
dy <- y - ybar
g <- expand.grid(dy, dy)
yiyj <- g[,1] * g[,2]

# make a matrix of the multiplied pairs
pm <- matrix(yiyj, ncol=n)

# multiply by the weights to se zero value for non-adjacent pairs
pmw <- pm * wm

# sum the values
spmw <- sum(pmw)

# divide by the sum of the weights
smw <- sum(wm)
sw  <- spmw / smw

# compute the inverse variance of y
vr <- n / sum(dy^2)

# last step to get Moran's I
MI <- vr * sw
MI
```

```
## [1] 0.1226804
```

How does our computed value of Moran's I compare to the expected value?

```
# expected value of Moran's
EI <- -1/(n-1)
EI
```

```
## [1] -0.125
```

Now, use the `spdep` package to calculate Moran's I.

```
library(spdep)
```

```
## Loading required package: sp
```

```
## Loading required package: spData
```

```
## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`
```

```
## Loading required package: sf
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

```
ww <- mat2listw(wm, style = 'B') # binary

# calculate value
moran(y, ww, n=length(ww$neighbours), S0=Szero(ww)) # K is the sample kurtosis
```

```
## $I
## [1] 0.1226804
##
## $K
## [1] 1.575566
```

```
moran.test(y, ww, randomisation=FALSE)
```

```
##
##  Moran I test under normality
##
## data:  y
## weights: ww
##
## Moran I statistic standard deviate = 1.943, p-value = 0.02601
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##         0.1226804        -0.1250000         0.0162500
```

We can also do a permutation test to evaluate the rank of the observed statistic in relation to the statistic of simulated values. In other words, we can take random "shuffles" of our data, calculate the Moran's I for each random shuffle, then compare the actual Moran's I to the random Moran's I.

```
moran.mc(y, ww, nsim=200)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  y
## weights: ww
## number of simulations + 1: 201
```

```
##
## statistic = 0.12268, observed rank = 192, p-value = 0.04478
## alternative hypothesis: greater
```

Question: What is the maximum number of permutations that we can do? https://bit.ly/324F9Ba

Now, make a Moran scatter plot to visualize spatial autocorrelation.

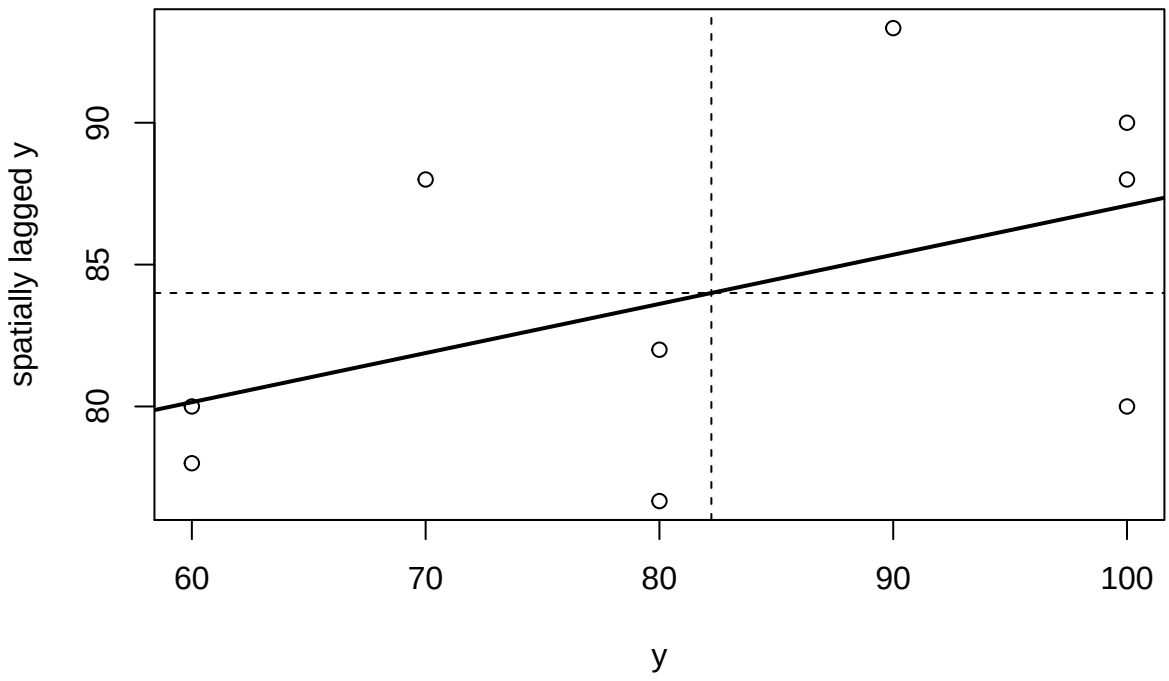
```
# get the neighboring values for each value
n <- length(y)
ms <- cbind(id=rep(1:n, each=n), y=rep(y, each=n), value=as.vector(wm * y))

# remove the zeros
ms <- ms[ms[,3] > 0, ]

# compute the average neighbor value
ams <- aggregate(ms[,2:3], list(ms[,1]), FUN=mean)
ams <- ams[,-1]
colnames(ams) <- c('y', 'spatially lagged y')
head(ams)
```

```
##     y spatially lagged y
## 1  90           93.33333
## 2  80           82.00000
## 3  60           80.00000
## 4 100           88.00000
## 5 100           80.00000
## 6  60           78.00000
```

```
# plot
plot(ams)
reg <- lm(ams[,2] ~ ams[,1])
abline(reg, lwd=2)
abline(h=mean(ams[,2]), lt=2)
abline(v=ybar, lt=2)
```

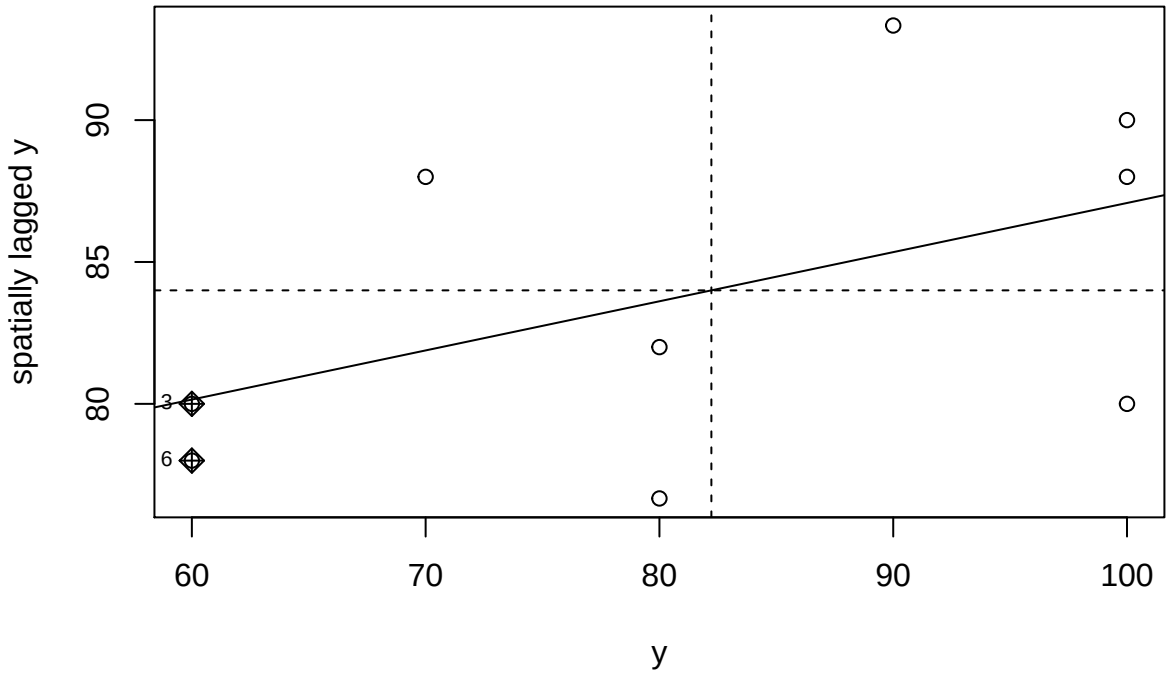


```r
# slope of the line
coefficients(reg)[2]
```

```
##  ams[, 1]
## 0.1731959
```

The package `spdep` can do the same thing.

```r
# now use the package to do the same thing
rwm <- mat2listw(wm, style='W')
moran.plot(y, rwm)
```



For more information, check out this link.