Moran's I

Generate the data

First, input our weights matrix w and our observations x.

```
# weights matrix (from the slide)
w <- 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))

# test scores (from the slide)
x <- c(90, 80, 60, 100, 100, 60, 100, 70, 80)
```

Calculate Moran's I

```
n <- length(x)
xbar <- mean(x)

# (xi-xbar)(xj-xbar) for all pairs
dx <- x - xbar
g <- expand.grid(dx, dx)
xixj <- g[,1] * g[,2]

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

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

# sum the values
spmw <- sum(pmw)

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

# compute the inverse variance of x
vr <- n / sum(dx^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
```

To compare our observed value of Moran's I to the expected value of Moran's I, we perform a hypothesis test. One way to do hypothesis testing is to calculate the z-score. The z-score looks like this:

$$z = \frac{I - E(I)}{var(I)}$$

Unfortunately, the variance of I is difficult to calculate. Check out the Wikipedia page to see for yourself. Instead, we will use the **spdep** package to calculate Moran's I and perform the hypothesis test.

```
library(spdep)
ww <- mat2listw(w, style = 'B') # binary

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

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

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

```
##
##  Moran I test under normality
##
## data:  x
## 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(x, ww, nsim=100)
```

```
##
##  Monte-Carlo simulation of Moran I
##
## data:  x
## weights: ww
```

```
## number of simulations + 1: 101
##
## statistic = 0.12268, observed rank = 93, p-value = 0.07921
## alternative hypothesis: greater
```

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

Plot spatial autocorrelation

Next, we'll make a Moran scatter plot to help visualize spatial autocorrelation.

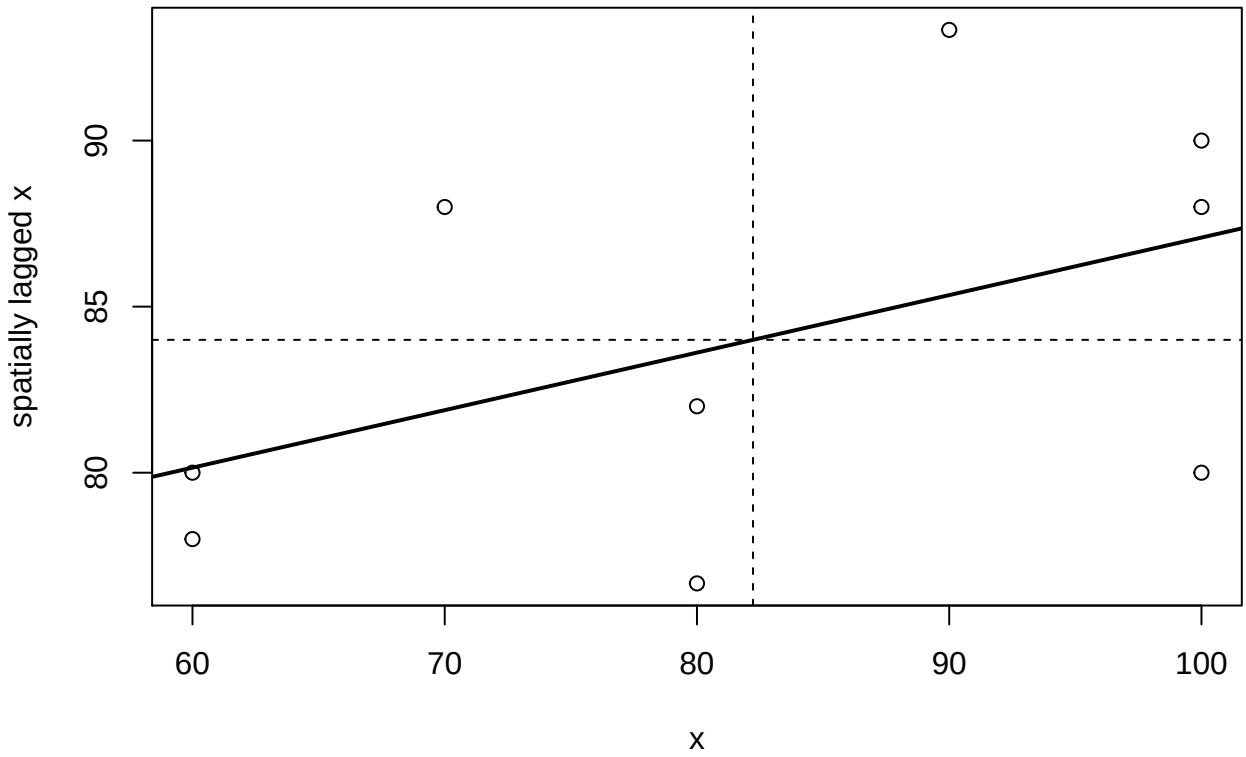
```
# get the neighboring values for each value
n <- length(x)
ms <- cbind(id=rep(1:n, each=n), x=rep(x, each=n), value=as.vector(w * x))

# 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('x', 'spatially lagged x')
head(ams)
```

```
##     x spatially lagged x
## 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=xbar, lt=2)
```

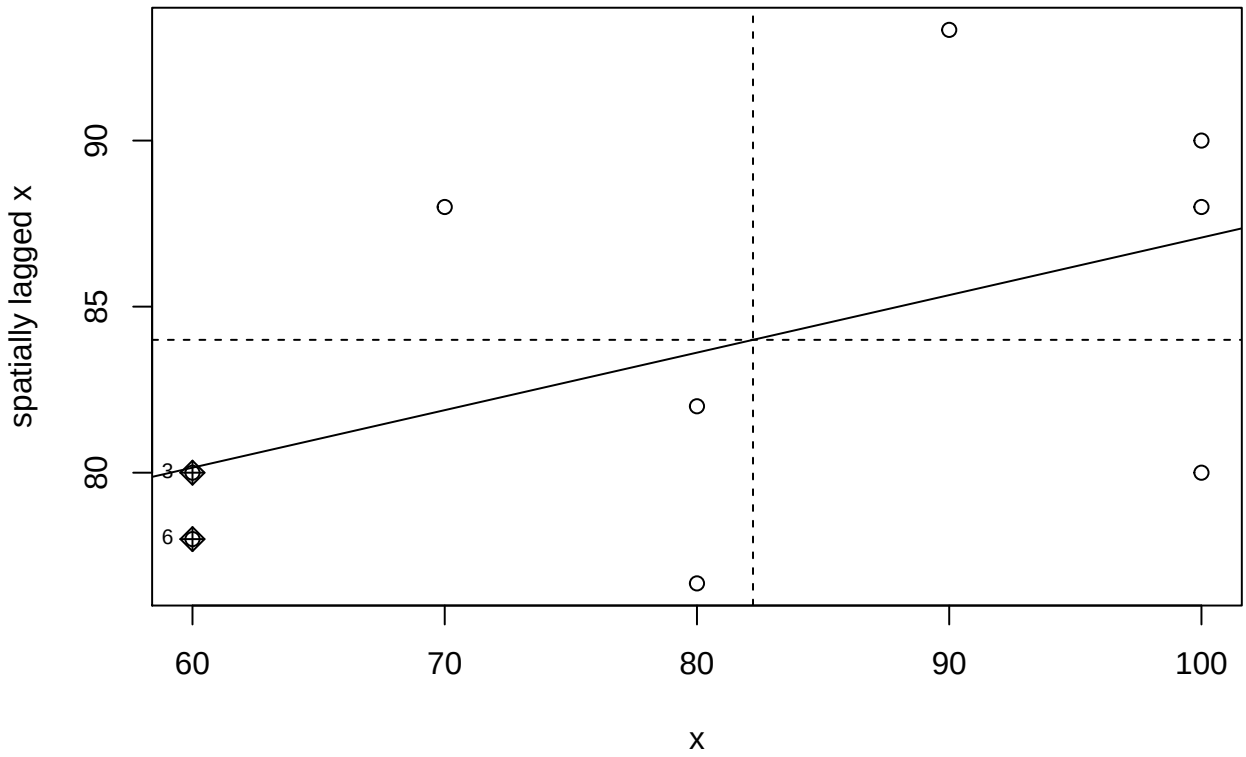



```
# slope of the line
coefficients(reg)[2]
```

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

The package **spdep** can do the same thing.

```
# use spdep to make a Moran plot
rwm <- mat2listw(w, style='W')
moran.plot(x, rwm)
```



For more information on spatial regression models in R, check out this link.