Join Count and Autocorrelation Analyses in R

Michael L. Treglia

Material for Assignment 6 of Landscape Analysis and Modeling, Spring 2016

This document, with active hyperlinks, is available online at:http://mltconsecol.github.io/TU_ LandscapeAnalysis_Documents/Assignments_web/Assignment06_Autocorrelation.html

Due Date: Tuesday, 8 March 2016

PLEASE WRITE YOUR NAME ON YOUR ANSWER DOCUMENT

Questions (Worth 1 point each, for a total of 10 points)

1) Convert the point data to rasters of presence/absence for *Acer rubrum*, *Pinus strobus*, and *Prunus serotina* as explained below (in the section 'Creating Cells of Presence/Absence'), and plot them. Based on the plots, for which species do you expect to find significant aggregation? For which species do you expect to find significant dispersion? Justify your answers.

2) Calculate the join count statistics for each species using the 'rook' type of joins. Which are actually significantly associated and significantly dissasssociated? (Remember, the alternative hypothesis 'greater' tests for significant association (the number of like joins is significantly greater than expected by random chance); the alternative hypothesis 'less' tests for the dissociation.)

3) Do the same analyses as in Question 2, but with the queen style setup. Do you see differences between these results and the results for Question 2 for any of the three species? If so how might you interpret it? (For example, does significance in one case perhaps indicate a more uniform distribution of an individual species?)

4) Create and show plots for Moran's I and Geary's c for *Acer rubrum*. Does this species show significant autocorrelation? If so, at what distances, and positive or negative? To speed processing, I recommend using a sample of only ~300 points.

5) Create and show plots for Moran's I and Geary's c for *Pinus strobus*. Does this species show significant autocorrelation? If so, at what distances?

6) There will likely be significant autocorrelation in at least one of these datasets (if not, just think about this hypothetically) - given autocorrelation at specific scales and the assumption of independent samples for statistical tests, how could you adjust your sampling scheme to avoid problems with autocorrelation and ensure measurements are independent??

7) Calculate Local Moran's I for a raster of the number of trees for *Acer rubrum*. Look at plots of those results and plots of the original data - are there distinct 'hotspots' of tree of this species? Circle them on your answer sheet (you can copy/paste the a result figure into Word and use drawing tools there to simply draw a circle around them). Hotspots are local areas that have higher number of trees within a cluster of a few pixels, and are indicated with higher values for the local indices.

8) Do the same as for Question 7, but this time for *Pinus strobus*.

9) Explain a potential application of join count analyses in one of your own datasets. If you do not have a relevant dataset, feel free to think of a hypothetical scenario you might be interested in.

10) Autocorrelation poses a problem in statistical analyses because close samples may not be independent of one another, violating assumptions of statistical tests. Explain two potential causes of autocorrelation. It might help to provide potential reasons for autocorrelation in a sample dataset (the tree dataset we use here would be fine).

Introduction

Join count and autocorrelation statistics are valuable in understanding spatial dependencies among sample units. This lab focuses on using R to calculate join count statistics, Moran's I, Geary's c, and some local autocorrelation statistics.

We will be working with the multi-year tree census dataset from Harvard Forest, the Lyford Mapped Tree Plot Data, available at http://harvardforest.fas.harvard.edu:8080/exist/xquery/data.xq?id=hf032, which we have used in previous analyses.

Though the dataset is a complete census, it is rare to have such a dataset due to logistical and financial constraints (e.g., funding, person-power, time, etc.). For the join count analyses, we will focus on all of the data (restricted to a portion of the study area), but we will only consider presence/absence within grid cells. For autocorrelation analyses, we will use all data in an area, but the same analyses could be done with a sample of the data, collected in systematic or random sampling schemes.

Though we will use R, calling on packages that aid in processing and analyzing spatial data, PASSaGE (used in Lab 4) also supports some of these analyses, and students are encouraged to explore PASSaGE as an alternative if they are curious.

Necessary Packages

The packages we will use in this lab are 'spdep', 'raster', and 'pgirmess'. See previous labs for details on how to install and load packages (the respective commands are 'install.packages' and 'library')

Importing and Sampling the Data

You can download the dataset, store it locally and load it into R, as you did in Lab 3, or as in Lab 4, you can import this dataset directly from the web using the code below. We'll import the data and assign it to 'HTrees'.

```
#use 'setwd([File Path])' to set your working directory.
HTrees <- read.csv("http://harvardforest.fas.harvard.edu/data/p03/hf032/hf032-01-tree.csv")
```

For the Join Count analysis we will focus analyses on *Acer rubrum* (red maple) and *Prunus serotina* (black cherry); for the autocorrelation metrics, we will focus on *A. rubrum*, *P. serotina* and *Pinus strobus* (white pine). For both types of analyses, we will conduct the analyses separately for each species, so we need to subset the data appropriately. For all analyses, we will also need the positions of the trees (variables 'xsite' and 'ysite'), and our focal variable for autocorrelation analyses will be the diameter at breast height in 1991, so we want to exclude observations with no data for that variable.

```
# Create subset of data for Acer rubrum
acru <- subset(HTrees, species == "ACRU" & dbh91 != "NA", select = c("xsite",
    "ysite", "dbh91"))
# Create subset of data for Prunus serotina
prse <- subset(HTrees, species == "PRSE" & dbh91 != "NA", select = c("xsite",
    "ysite", "dbh91"))
# Create subset of data for Pinus strobus
pist <- subset(HTrees, species == "PIST" & dbh91 != "NA", select = c("xsite",
    "ysite", "dbh91"))
```

Creating Cells of Presence/Absence (For Join Count Analysis)

As discussed in lecture, join count analysis is used to detect aggregation of like categories among sample units. Sample units can vary, and be in the form of polygons, points, or a lattice of grid cells. For this example, we will focus on a grid, created for the focal area of the Harvard Forest tree dataset. There are a few different ways to do this, but we'll create a raster layer for a core area in the study plots, and assign pixels 1 or 0 for presence or absence of individual species, respectively.

First, we'll load the required packages, and convert the points to SpatialPointsDataFrames.

```
# Load required packages
library(raster)  #the 'sp' packages gets loaded with these, so no need to load it separately
```

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

```
library(spdep)
```

```
## Loading required package: Matrix
```

```
library(pgirmess)

# Convert tree data to SpatialPointsDataFrame, both for entire dataset, and
# for individual species
HTrees.spdf <- HTrees
coordinates(HTrees.spdf) <- c("xsite", "ysite")

pist.spdf <- pist
coordinates(pist.spdf) <- c("xsite", "ysite")

acru.spdf <- acru
coordinates(acru.spdf) <- c("xsite", "ysite")

prse.spdf <- prse
coordinates(prse.spdf) <- c("xsite", "ysite")
```

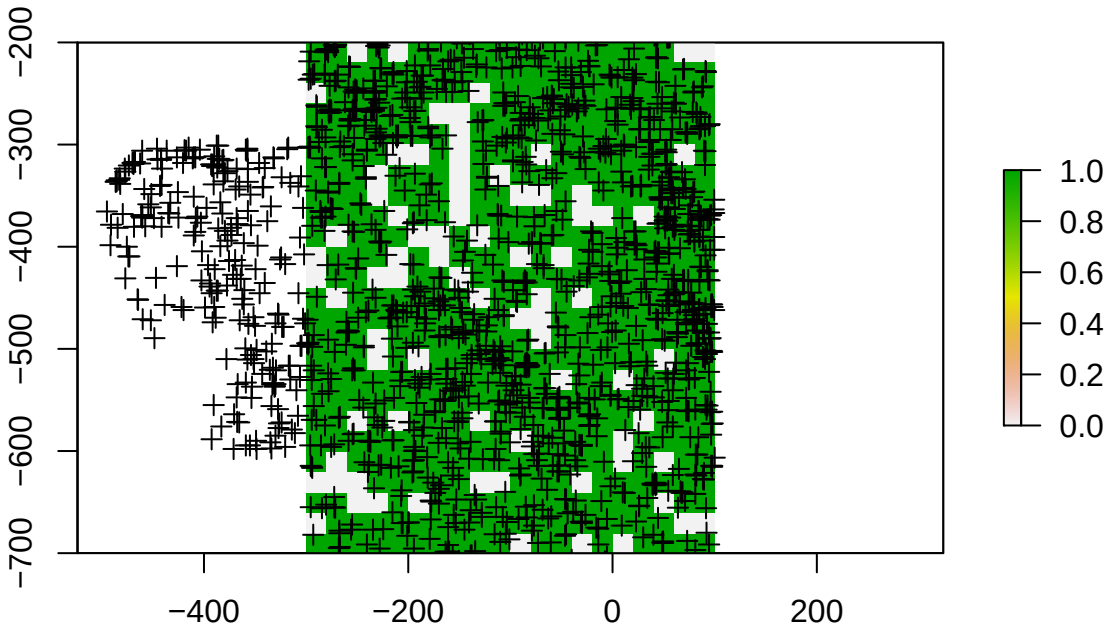
Now you can plot these datasets using the 'plot' command as you have done in previous labs; to view the x and y axes, use the argument 'axes=TRUE'. Looking at the full dataset, you can identify the extent of the study area. For these analyses, we will simply select a rectangular area that includes areas only areas that were sampled for trees, bounded by the coordinates indicated below. The function we will use to define the extent is 'extent'. Then, we will create a blank raster with 20 foot grid cells (20 cells in the x-dimension, and 25 cells in the y dimension).

```
#Define the extent for the join count analyses
jc.extent <- extent(-300,100,-700,-200)
#set up a blank raster
r <- raster(nrows=25, ncols=20, ext=jc.extent)
```

Next, we will rasterize the points, to the same grain size and extent as the blank raster we just established. The command for this is 'rasterize'; we will use the argument 'field=1', which indicates that if points are present in a grid cell, the cell will take a value of 1. If the cell contains no cells, it will remain a NoData cell, thus the subsequent line sets all remaining NoData cells to 0 (indicating absence).

```
acru.rast <- rasterize(acru.spdf, r, field = 1)
acru.rast[is.na(acru.rast)] <- 0

# You can plot the result
plot(acru.rast)
# You can plot the points on top of the raster to verify it is correct
# Remember - we only mae the raster for a subset of the data, so there will
# be points in a larger area than the raster covers
plot(acru.spdf, add = TRUE)
```



Follow the example above and do the same for the other species (plotting is up to you)

Setting Up and Running Join Count Analysis

For we need to define the neighbors - this is the connectivity matrix that was discussed in lecture.

```
# Generate neighbors list - the function is 'cell2nb' and the arguments are
# the number of rows and colums in your grid; you can simply get those
# characteristics from your any of your rasters using 'nrow' and 'ncol'
# commands, nested in the cell2nb function (as ilustrated below). Note, the
# default for this is 'rook', but you can change the join counts to 'queen'
# by adding the argument 'type='queen'.
nb <- cell2nb(nrow = nrow(acru.rast), ncol = ncol(acru.rast))
# To calculate neighbors for queen configuration
nb.queen <- cell2nb(nrow = nrow(acru.rast), ncol = ncol(acru.rast), type = "queen")

# Convert the neighbors list to a 'weights' list; again, this will be the
# same for all species we are analyzing. You an follow the example below
# using 'style='B' (as a Binary weights matrix). Again, calculate this for
# the queen setup as well as the default (rook) setup.
lwb <- nb2listw(nb, style = "B")
lwb.queen <- nb2listw(nb.queen, style = "B")
```

Now, everything is set for the analyses. You have a raster,which has the values of presence/absence for species in the grid cells, and the weights information (as object 'lwb'). The actual values for the pixels are in 'slots' of the rasters and as you can see by looking at the structure of these datasets. To view the pixel values, you can all the 'data' slot. For example

```
acru.rast@data@values
```

Finally, the functions to run the join count test are 'joincount.test' and 'joincount.mc'. The first option calculates a p-value for the test using a z-statistic, assuming a normal distribution, whereas the second is for a permutation test, based on random permutations of the dataset. Given that this test is designed for categorical data, it expects the presence/absence data as a factor, which can be done within the functions. Another thing to note is that you can select the alternative hypothesis (the null hypothesis being that the joins among presence and absence cells are random). The default is 'greater', which is testing the alternative hypothesis that the number of like joins is more than expected by random chance. changing this to 'less' would switch the hypothesis being tested to represent that the number of like joins is fewer than expected by random chance (indiating higher levels of dispersion). Again, the choice of rook or queen-based setups can be established earlier, using the cells2nb function. An important thing to remember is that the expected values are based on the proportions of presence and absence cells in the actual data.

```
# First, the regular join count test for Acer rubrum (Testing the hypothesis
# of aggregation among like categories; add the argument 'alternative='less'
# to reverse this)
joincount.test(as.factor(acru.rast@data@values), lwb, alternative = "greater")
# Second, the permutation-based jon count test; similar to above, and you
# can adjust the number of simulations with the 'nsim' argument
joincount.mc(as.factor(acru.rast@data@values), lwb, nsim = 999, alternative = "greater")

# Can also compute these for the queen setup; for example, with the
# permutation test:
joincount.mc(as.factor(acru.rast@data@values), lwb.queen, nsim = 999, alternative = "greater")
```

The join count statistics are calculated and presented for '0' and for '1' in this case, as those are the factor levels of interest (where 1 is presence, 0 is absence). Because you are focused on

Analyzing Global Moran's I and Geary's C

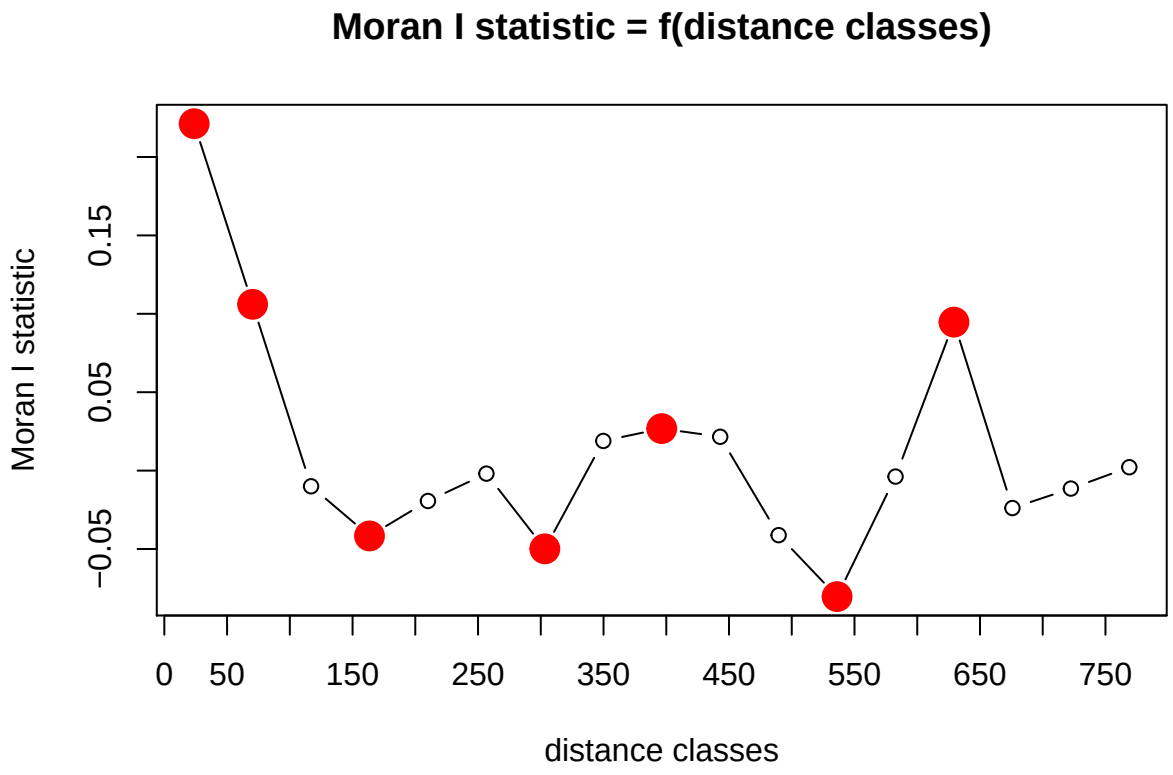
As discussed in class, we frequently want to quantify autocorrelation for single variables across multiple samples. For these exercises, we will focus on the variable 'diameter at breast height' for *P. strobus* and *A. rubrum* from 1991 - each of which have enough individuals that we can likely calculate these statistics reasonably well for a variety of lags (i.e., distance classes). These can be slow to compute for large sample sizes in R; given that we have ~1800 observations of *A. rubrum*, we will run the analyses on a smaller, random sample of only 300 points (without replacement) for that species:

```
# Use the function 'sample' to get random sample of points; this is done on
# a SpatialPointsDataFrame , but could also be run on a regular dataframe
acru.sample <- acru.spdf[sample(1:nrow(acru.spdf), 300, replace = FALSE), ]
# plot(acru.sample) #plot if you want to see waht the random sample of data
# looks like
```

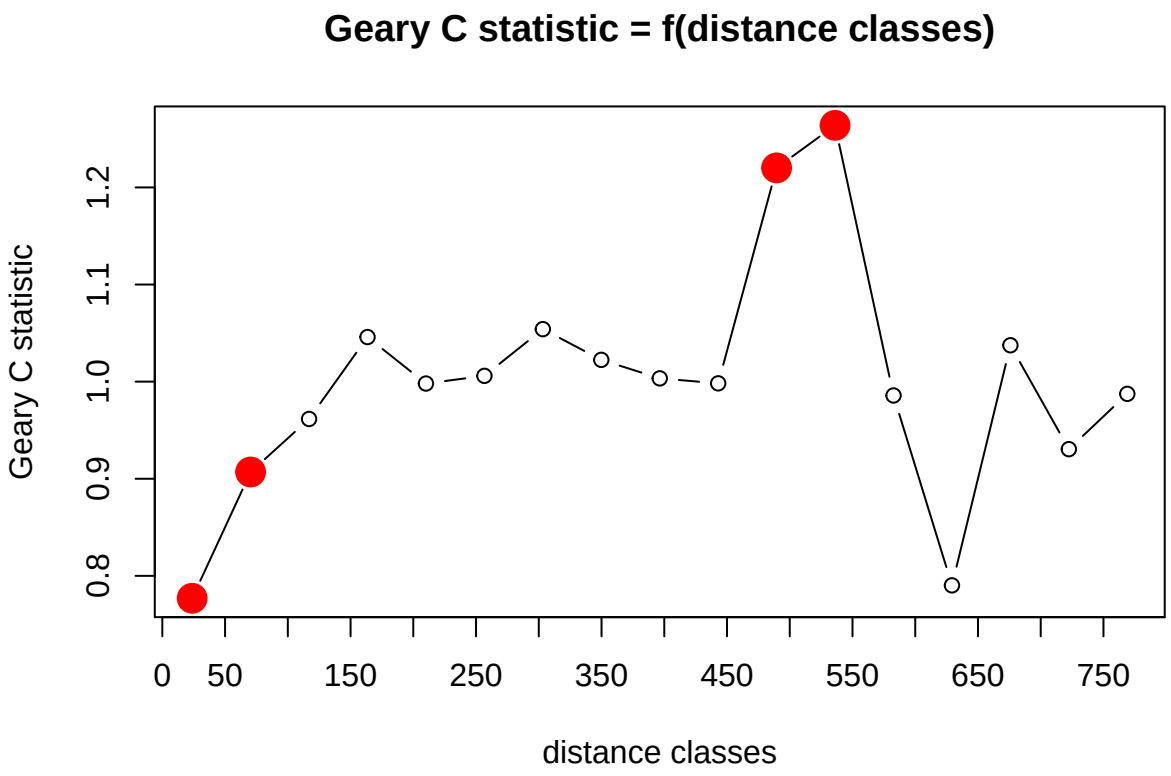
Moran's I and Geary's c can be calculated using the 'spdep' package, with lots of customizability. For example, you could calculate these metrics for very specific distance classes, whereas some other packages set the distance classes automatically. As with the calculating join count statistics however, it requires setting up

your own neighbor matrix, then we would need to figure out distance classes and . For this lab, we will use the 'pgirmess' package, and the function 'correlog'. This by default uses Sturge's rule for setting the number of distance classes, and equal-sized distance classes. You can switch between Moran's I and Geary's c using the 'method' argument. Plotting the result will automatically indicate significant values as red dots; the basic output shows upper limit of each distance class, the Moran or Geary coefficient,the p-value, and the number of pairs of points in that distance class. Below is the code for *Acer rubrum*; you should do this for *Pinus strobus* on your own.You do not need to do this for *Prunus serotina* as there are simply too few points. If you look up the help for these functions, you will see it takes arguments from moran.test() and geary.test(). By default, these functions are testing for positive autocorrelation, but we can use the 'alternative' argument to change that - you can use 'two.sided' to test for both positive and negative autocorrelation, 'greater'(which is the default) for positive autocorrelation, or "less" for negative autocorrelation.

```
acru.moran <- correlog(coordinates(acru.sample), acru.sample$dbh91, method = "Moran",
    nbclass = NULL, alternative = "two.sided")
# Can view the textual results by simply typing 'acru.moran'
plot(acru.moran)  #plots the results
```




```
acru.geary <- correlog(coordinates(acru.sample), acru.sample$dbh91, method = "Geary",
    alternative = "two.sided")
plot(acru.geary)
```



Calculating Local Moran's *I* statistics

As with the global autocorrelation statistics, a number of packages can be used to compute local autocorrelation statistics. We will focus on looking for 'hotspots' of high densities of *Acer rubrum* and *Pinus strobus* in the Harvard Forset Tree data using Local Moran's *I*, calculated in the 'raster' package. This can also be done for various types of spatial datasets using the 'spdep' package.

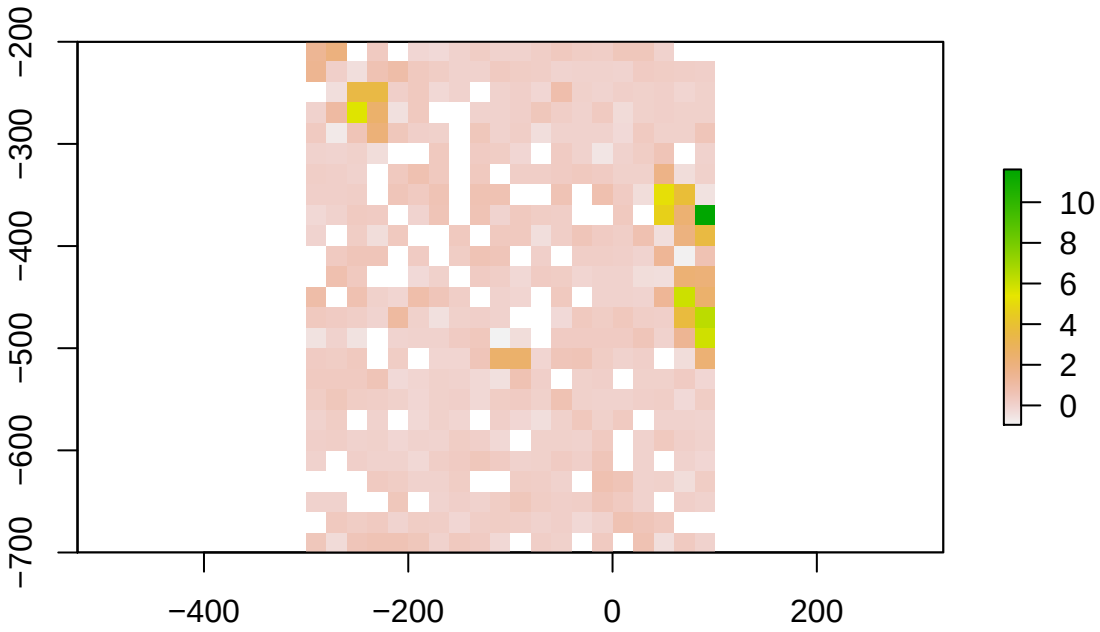
We will use the same general techniques to create rasterized datasets of the tree datasets as we did for calculating Join Count Statistics, but instead of simply presence/absence, we will calculate the number of trees per grid cell. We will replace the argument 'field=1' with a function, that calculate the 'length' of the dataset (the number of values) within each raster pixel.

Note - we put [[1]] after the function to indicate that we only want the first layer - the function actually calculates two layers - one for the 'ID' value from the Spatial Points Data Frame and one for the dbh91 values. Since these are simply counts of the number of points, both layers will look the same and we could choose either one. If we did not use the '[[1]]', and went to plot the result, you would see two rasters plotted. This makes an important difference in some cases, for example if you are using 'fun=mean', which calcualtes the average value of fields for all points within a pixel - you can explore and see what happens.

```
acru.rast.count <-rasterize(acru.spdf, r, fun=function(x,...)length(x))[[1]]
```

Now we can calcualte a local Moran's *I* for the raster, and plot the results. The function for this in the 'raster' package is simply 'MoranLocal' - check out the options you can adjust; importantly, you can change what surrounding pixels it uses for this calculation, thus you could calculate this using a 'rook' or 'queen' setup, as you could with the join count statistics. Here is the example for *Acer rubrum*:

```
plot(MoranLocal(acru.rast.count))
```


−200
−300
−400
−500
−600
−700
−400
−200
0
200
10
8
6
4
2
0