Week 5 Computer Lab Introduction: Spatial Statistics

Scene 1: Spatial Statistics

Hello again! This week, we'll do spatial analysis with the snail data from last week. Here we have a map of the sites, where the color indicates the rarefied allelic richness. Some sites in red have high richness, others in blue have low richness.

The main goal today is to test for spatial autocorrelation in the snail data set. We will do so at two analysis levels, first for the genotypic data based on pairwise genetic distances among populations, and then for the site-level variables, such as genetic diversity, site-level FST and predictor variables.

This video focuses on the technical challenges: how do we get from allele frequencies to genetic distances, and how are these represented in R? We'll see how to do an exploratory analysis of the pairwise distances and discuss three ways to analyze them. Then we'll learn how to define neighbors and spatial weights in R with the package 'spdep', and how to extract attribute data when the results of an analysis are returned as an S3 or S4 object.

Scene 2: Genetic Distances

To calculate pairwise genetic distances, we first need to convert the genotypic data to allele frequencies. Here we start with a single microsatellite locus A for four individuals of a diploid species. In this example, locus A has three alleles, 204, 210 and 218. The numbers actually refer to the tandem repeats of a specific motif. The first individual has one allele 204 and one allele 210. Here we use relative frequencies, so the first individual gets 0.5 for 204 and 0.5 for 210, and 0 for 218. The second individual has only the allele 204, so that's a 1 and two 0's. Individual 3 has alleles 210 and 218, and individual 4 has missing values for this locus.

When we use the function 'makefreq' from the 'adegenet' package to calculate these frequencies, we have three options how to treat missing values. Here, they were counted as 0's, but we could specify that they should be represented as NA's in this table, or replaced by the column mean. Which option to use depends on what analysis we want to do later on.

Now we can calculate the proportion of shared alleles. Individuals 1 and 2 have one allele in common out of two, so that is 50 %. Here we have a square matrix, where the rows and the columns both represent the individuals. We compared the second individual to the first, hence we write 0.5 into the second row and the first column. We get the same value if we compare the first individual to the second one, so we write 0.5 also into the first row and the second column. On the diagonal, we would compare each individual to itself, so we would get 100% shared alleles. Now we can enter the rest of the values.

The proportion of shared alleles tells us how genetically similar two individuals are. Genetic distance, on the other hand, tells us how different they are. To convert the proportion of shared alleles to a measure of genetic distance, 'Dps', we subtract it from the maximum, 1. Now we have zero's on the diagonal, and the larger the values, the more different two individuals are.

If we have sampled individuals from discrete patches that we treat as local populations, then we should calculate genetic distances among populations, not among individuals. For that, we aggregate the data to get allele frequencies for each population. These can take many values between zero and one, but they should again sum to 1 across all alleles of the same locus. There are many different measures of genetic distance in different R packages for genetic analysis, for example in 'adegenet' or 'gstudio'. Each returns a square matrix where the rows and columns represent local populations. The diagonal values are zero, and the off-diagonal values are the pairwise genetic distances.

There is one caveat here. Landscape genetic data often have missing values, and how these are treated can affect the results. It is important to check along the way how the missing values are treated, when you import the data, when you calculate the allele frequencies and the genetic distances, because the behavior at one step may depend on the treatment at an earlier step.

Scene 3: 'dist' Objects in R

R has a special object class for pairwise distances, 'dist' objects. Distances can be calculated for any quantitative variable, not only allele frequencies, and we'll use the generic method to calculate geographic distances between sampling locations. We start with a table of x and y coordinates. Now these coordinates must be in a metric coordinate system, such as UTM coordinates. If they are lat-long coordinates, they need to be converted first. Then we calculate either Euclidean distance or great-circle distance. The difference matters most if we have a large study area, where the curvature of the earth starts playing a role. Euclidean distance calculates the shortest distance in a plane, two-dimensional space, whereas the great-circle distance calculates the distance along a sphere that approximates the spherical shape of the earth.

For calculating Euclidean distance, we can use the generic function 'dist'. It returns and object 'D' of class 'dist', which we can access it in three different ways. With 'as.matrix', we get the full distance matrix, including the diagonal. With 'as.dist' we get a sparse representation that shows only the lower triangle of the full distance matrix. This is enough because the values are symmetric anyways, and the diagonal is not meaningful. We can also extract the values from the lower triangle of a 'dist' object with the function 'as.vector'.

Scene 4: Exploratory Analysis

Before we do any statistical tests, we should explore the data visually to get a feel for what is in the data. We can plot the genetic distances against the geographic distances. Here, each point represents a pairwise comparison between two populations, and the line indicates a linear regression. In this example, the line is horizontal, which suggests that there is little relationship between genetic and geographic distances.

If we have many populations, or if we do this at the individual level, we can get thousands of points and it becomes hard to see a pattern. We can calculate a two-dimensional kernel density to visualize the density of points in the scatterplot. Here, red is a high density of points, yellow intermediate, and blue low. The red line is a smooth regression line, which takes a moving average. This can show non-linearities. Here for example, we have a slight increase of genetic distance for short distances, and a decrease for large distances. So if we had sampled a much smaller study area, we might have found a positive relationship.

Let's look at the pseudo-code for creating this figure. First, we calculate the kernel density from the geographic distances 'Dgeo' and the genetic distances 'Dgen', and store it as 'density'. Then we define a color palette that goes from white to blue to gold to orange to red. Then we use the function 'image' to plot the density over the scatterplot. Finally, we fit a smooth regression line with the function 'loess.smooth'.

Scene 5: What hypothesis to test?

We want to test for isolation-by-distance, that's our biological hypothesis. In spatial statistics, this means that we expect to find positive spatial autocorrelation, where nearby samples are more similar than distant ones.

As with any test, we should not only consider the p-value but three aspects of the result. The direction of the effect here means whether we have positive or negative spatial autocorrelation, the size of the effect refers to the strength of autocorrelation, and the statistical significance refers to the p-value of the test.

Mantel test: But what test? There are three that are commonly used. The first is the Mantel test. This is simply the correlation between the genetic and geographic distances. Because we are testing for IBD, it makes sense to use a one-sided test.

In population genetics, we can get further interpretation if we plot linearized Fst against the log of geographic distance. Linearized Fst is Fst divided by one minus Fst, and in a two-dimensional data set, we expect a roughly linear relationship with the natural logarithm of geographic distance. If that is the case, and some assumptions are met, then the regression slope b is one over four times sample size N times pi times sigma squared, the variance of dispersal distances.

Correlogram: Another approach is to calculate a correlogram. Here we have a Mantel correlogram. The geographic distances are binned into distance lags, and each dot here represents the Mantel correlation among all pairs in the same distance lag. We can test the value of each distance class whether it is different from zero. If we are testing for IBD, it makes sense again to use a one-sided test. Also, we expect the strongest autocorrelation in the first lag, then a levelling off towards zero. Testing the second lag only makes sense if the first lag was significant. This is called sequential testing, where we stop after the first non-significant lag. This means that we don't know beforehand how many tests we will perform. Therefore, we use a sequential Holm's correction, where the second p-value is adjusted for two tests, the third p-value for three tests, etc.

When we do a correlogram for a single variable, we don't use a Mantel correlogram but a Moran correlogram, but the rest remains pretty much the same.

Moran's I: Moran's I is related to a Moran correlogram, but we usually calculate it differently. Instead of distance lags, we calculate it for first, second, third etc. neighbors. We'll get to that in a moment. This makes sense if we think of a stepping-stone model, where dispersal happens only between neighboring sites, but sites that are further apart can be linked by multiple steps. Again, it makes sense to do a one-sided test, and we expect that the first value is the highest and then a tapering off towards zero.

However, in a stepping-stone model, the observed autocorrelation between second neighbors A and C may be explained by the sum of the two steps between first neighbors A and B, and B and C. If we would partial out the effect of first neighbors, then very likely the autocorrelation among second neighbors would not be significant anymore. Therefore, we usually calculate and test Moran's I only for first neighbors.

Scene 6: Three Analytical Paradigms

Why do we have three different tests for essentially the same thing? These tests come from different disciplines, reflecting different analytical paradigms. And in practical terms, they consider different subsets of the pairwise distances, and therefore they may lead to different results.

All pairs: The Mantel test comes from ecology, where we analyze the multivariate dissimilarity in species composition or allele frequencies based on all pairs. This results in single value for the Mantel correlation. The method is implemented in R packages 'ade4' and 'vegan', which focus on multivariate ecological data analysis.

Distance lags: The distance lag approach comes from geostatistics, from the analysis of random fields, which are quantitative variables that vary continuously across space and are best represented as raster surfaces. Here we get a value for each lag, and the focus is on the distance-decay function. We'll get back to this next week. The main goal of the package 'EcoGenetics' is to make it easy to do this kind of analysis with genetic data.

First neighbors: The analysis framework with spatial neighbors comes from geography, from the analysis of spatial objects that are best represented with vector data. Think of driving from one state to another. Adjacent states that share a border are first neighbors, that's where you can get directly without transit through another state. In this graph representation, each circle shows the center of a state, or node, and adjacent states are connected with a line.

Pretty much all methods for neighbors use the R package 'spdep', which stands for spatial dependence. To test Moran's I, the most robust option is moran.mc, and there is also a function specifically 'lm.morantest' to test for spatial autocorrelation in the residuals of a linear model.

Scene 7: Neighbour topology with 'spdep'

When we are not dealing with a complete polygon map but with habitat patches in a matrix, defining neighbors is less straight forward. We can define them manually or use an existing algorithm, like the minimum spanning tree, Gabriel graph, Delaunay triangulation. Or we can consider all sites as neighbors but give them different weights, so that nearby sites have a larger weight than distant ones.

'Adegenet' has a helper function, 'chooseCN', which stands for choose a connection network. How do we decide between these networks? In spatial statistics, the idea is that we predict the value at one site from the local mean calculated from its neighbors, and that this local mean is a better predictor than the global mean. With the minimum spanning tree, each site has only one or two neighbors. That's a bit low for calculating a local mean, we want more neighbors. If we use too many neighbors, we lose information because the local mean becomes too similar to the global mean. For many purposes, the Gabriel graph is a reasonable compromise.

Once we have defined the neighbors, we need to decide about the spatial weights, that is, how much weight to give to each neighbor when we calculate a local mean. Binary weights are 1 for neighbors and zero for non-neighbors. Here we have the binary weights for the Gabriel graph. In the first row, we have the neighbors of the first site, these are sites 2 and 3. Hence the row sum is 2. Site 2 has three neighbours, hence the row sum is 3. Most often, we row-standardize the weights, so that the row sums are 1. This is done with the argument 'style = W' in the function 'nb2listw', which means 'convert neighbours to a list of spatial weights'. If the sampling design is very irregular, we may want to further modify the weights based on the actual geographic distances between neighbors. Technically that's a bit more challenging.

Scene 8: Object Type in 'spdep'

The package 'spdep' has two main object classes, where 'nb' defines the neighbours and 'listw' is a list of spatial weights.

'nb' is a list of length n, one for each population. Each list element contains a vector with the row numbers of the neighbours of that population. Again, we see that the first population has two neighbours, populations two and three. These are row numbers, not names.

An object of type 'listw' is a list with three elements. The first just defines the method, the second holds an 'nb' object that defines the neighbours, and the third is a list with the spatial weights. Here we have row-standardized weights, where both neighbors of the first population, that is sites 2 and 3, have the same weight of 0.5.

Scene 9: Extracting Results

Finally, no matter which test we use, we'll want to extract the results. Each test returns an object with lots of information. We can print it, but sometimes we want to extract some values, like a p-value. This is especially important when we run the same analysis many times.

For one genetic distance matrix: Let's return to the Mantel correlogram. If we apply the function 'eco.cormantel' to a single genetic distance matrix, the result is an object of class 'eco.correlog' that contains a lot of information in different slots. This is an S4 object, and we use the function 'slotNames' to check what slots it has. For an S3 object, we would use the function 'attributes' to get the attribute names.

The main slot of an eco.correlog object is called OUT, in capital letters. We can use the 'at' symbol to access a slot of any S4 object. In addition, the EcoGenetics package has a type of function for accessing slots, 'ecoslot' dot - and the name of the slot, which is supposed to be safer and has additional functionality. Let's use it here to store the output in an object 'Result'.

This slot is actually a list with a single element. A list is an S3 object and we can use the square brackets to access its elements. One thing that is confusing about lists is that we an use single or double square brackets. Both work, but they give different results. Normally, we use the double brackets, which extract only the list element itself. In this case, this is a matrix.

What looks like the first column is actually the row names. You can tell this because there is no column name. The row names here contain the definition of the distance lags, so we see that the first distance lag contains distances between 0 and 4228 meters. The first column 'd.mean' contains the mean distance of the pairs in this lag, 2090, the second column 'obs' contains the observed value of the Mantel statistic for each distance lag, the third column contains the expected value under the null hypothesis, the fourth column contains the p-values with sequential Holm's correction, and the last column 'cardinal' contains the number of unique pairs in each lag. Typically, the last few lags contain few pairs and should not be interpreted.

Let's say we want to extract the observed value for the first distance lag. We just add another set of single square brackets to index the elements of the matrix. What we want is the first row and the 2nd column. We separate the rows and the columns by a comma. In the same way, we extract the p-value of the first lag by selecting the first row and the fourth column.

Now we can put the code together. We access the slot 'OUT' of the object 'Res', take the first list element, which is the matrix, and from the matrix, we extract the first row and the second and fourth column. This will return two values, the observed Mantel statistic and the p-value, both for the first lag.

For a list of distance matrices: In this week's lab, we will calculate several genetic distance matrices and store them in a list, let's call it 'DgenList' here. Then we want to calculate a Mantel correlogram for each genetic distance matrix. We have already learned how to use 'lapply' to apply a function to each list element. 'lapply' returns a list with results, let's call it 'ResList'. Each list element is an S4 object of type 'eco.correlog'.

We can apply any function to the list elements of ResList again with lapply. Here, we want to extract only a vector with two values per list element, the observed value and the p-value for the first distance lag. This means that we can use 'sapply' instead of 'lapply', which simplifies the output. For the function of x, we use the same expression as for the single analysis, only that we replace 'Res' by 'x'.

With just two lines of code, we can thus test for IBD with separate Mantel correlograms for different genetic distance measures and extract the key results conveniently as a single table.

Scene 1: Spatial Statistics

This brings us to the end of this video. Now it is your turn to do some spatial statistical analysis in the interactive tutorial and eventually adapt the worked example to your own data as needed.