Week 4 Computer Lab Introduction: Metapopulation Genetics

Scene 1: Metapopulation Genetics

Hello again! For the next two labs, we'll analyze a new data set for a freshwater snail on the Caribbean island of Guadeloupe. The snail occurs in ponds like this one. The goal of this lab is to test for genetic evidence of metapopulation dynamics. First, we will quantify genetic structure at different hierarchical levels. Then we will test effects of population size and connectivity on genetic diversity and differentiation. Finally, we will use multi-year genetic data to test for effects of recent extinction events.

This video focuses on some of the methods used. First, I will introduce the 'EcoGenetics' package for integrating genetic, spatial and ecological data. We'll have a quick look at the analysis of molecular variance, and brush up how to interpret regression results and check regression assumptions. A second video deals with R graphics, especially the 'ggplot2' package.

Scene 2: Freshwater Snail Data

This example has been contributed by Thomas Lamy, who sampled 25 ponds, or sites, across the island. Some ponds were sampled as part of a cluster of nearby ponds to capture local gene flow patterns.

For each site, we have genetic data for 22 - 32 snails that were genotyped at 10 microsatellite loci. In addition to the raw data, we have derived genetic response variables that include rarefied allelic richness, expected heterozygosity, site-level FST that was calculated with the software GESTE, and temporal FST, which measures for one site the genetic differentiation between years. The predictor variables include long-term population size NLT, pond size, hydrological connectivity C, wetland density D within a radius of 2 km, and APE, which refers to whether or not the site appeared to have gone extinct over the course of the study, for example if the pond dried out completely.

We will perform three types of analysis with different subsets of the data. With the hierarchical data set that consists of 4 clusters times 3 ponds for a total of 12 ponds, we will test if ponds from the same cluster have more similar allele frequencies than ponds from different clusters. Technically speaking, we will use analysis of molecular variance, AMOVA, to compare genetic differentiation within and between clusters.

The spatial data set contains all 25 sites, and we will use it to test whether genetic diversity increases with population size, and genetic differentiation decreases with connectivity, as expected from theory. We test this with multiple regression.

Finally, the temporal data set contains multi-year data for 12 sites, 5 that had an apparent extinction event during the study and 7 that did not. We will use a t-test to test whether the populations that experienced an apparent extinction event show higher differentiation between years, as we would expect after a recent bottleneck or recolonization from nearby ponds.

Scene 3: Package 'EcoGenetics'

The data are already available as an 'ecogen' object, which is an S4 object type from the new package 'EcoGenetics'. An 'ecogen' object has several slots to accommodate different types of data. Slot XY contains the spatial coordinates, slot P is empty here but is meant for phenotypic traits. Slot G contains the genotype, here the 10 microsatellite loci for all 1270 genotyped snails. Slot E contains the ecological site variables, and slot S the structure variables, that is, the information about which pond, cluster, year etc. each snail was sampled. Any other variables could be added to slot C. Two slots are filled automatically: slot A contains the allele frequencies, and slot OUT is reserved for output from functions that have been written specifically to analyze ecogen objects.

We can access each slot separately. There is a special function ecoslot, followed by a period and the slot name, to access a specific slot, or we can use double square brackets and the slot name in quotes. Please don't use the 'at' symbol for ecogen objects, it does work but it is not dependable because it does not give you error messages if something goes wrong.

Why should you use ecogen objects at all? They are really convenient for a number of reasons. First, you can easily combine and store data from multiple sources and ensure that they are always matched by row. Second, it is easy to import and export to and from other packages, including adegenet, gstudio, and commonly used population genetic software even beyond R. This means that you can use EcoGenetics to convert for example from gstudio to a genind object by first importing into an ecogen object and then exporting to a different format. Third, EcoGenetics has functions to calculate spatial statistics of your genetic or ecological data, we'll look at these next week.

A few notes on data management with EcoGenetics: we already learned that we can use double square brackets to access a slot. Single square brackets can be used to slice an 'ecogen' object, here for example we would create a new 'ecogen' object that contains only the first three individuals. Another important thing to know is that the sorting all goes by row names. So, to construct an 'ecogen' object, your table with the genotypes must have the same row names as your table with site variables and the table with the spatial coordinates, etc., and they all must have the same number of rows or you'll get an error message. But, let's say you have a table with site-level variables like patch size, and you want to extract patch size for each individual snail. You can do this with the function 'eco.fill_ecogen_with_df', which finds the relevant rows based on a column that you can specify. In the other direction, you can also aggregate the individuals from an 'ecogen' object to an 'ecopop' object, where each row is a population.

Scene 4: AMOVA

The first analysis we'll do in this lab is an analysis of molecular variance, AMOVA. This is like an ANOVA, but instead of a single response variable, we start with a multivariate table of allele frequencies. We use it to quantify genetic structure at different hierarchical levels. This is like the F-statistics where we had FST that measures differentiation among populations. FST has a specific interpretation in population genetic theory, but this interpretation is only applicable if a number of assumptions are met that may not hold for our data. In contrast, AMOVA does not make these assumption, it does not reference a population genetic model, but this also means it does not have a direct theoretical interpretation. Instead, with this AMOVA, we want to know how much variance is explained by clusters, how much by populations, and how much remains unexplained variance among individuals within populations. Then we want to test whether the variance explained at each level is significantly larger than zero. Finally, we get a measure, called Phi statistics, that are analogue to F statistics.

Where do we find this information in the AMOVA output? Here we see that cluster explained only half a percent of the total variance (a), populations within clusters explained 8.9% (b), and about 90% variance was among individuals within populations (c). This is a striking result, because this means that snails from nearby ponds are genetically just as different as snails from ponds far apart, so we don't really see a pattern of isolation by distance. Here we have the p-values, in reversed order. The variance within and among populations are statistically significant, but the p-value for cluster is right around 0.05, our significance level alpha. The test is done with permutations, and if we ran it a few times, it might be significant sometimes and sometimes not. In short, it is not sure whether or not we can rule out chance for explaining the observed structure at the cluster level.

Finally, here are the Phi statistics. We can derive them from the variance components a, b and c that were expressed as percent of total variance. Phi.ST is the total structure among sites and is calculated as the sum of a and b. Phi.SC is the structure among sites with the same cluster, and it is calculated as b divided by b plus c. Phi.CT is the structure among clusters, and this is the same as a.

Scene 5: Regression Interpretation

The next step is a multiple regression analysis. Here we try to explain variation in site-level FST by two variables, long-term population size NLT and hydrological connectivity C. Site-level FST measures how different the allele frequency in one population are from all other populations, and we would expect this to be high in small and isolated populations, where drift is high and gene flow is low. We use the function 'lm' to fit a linear model, then in the bracket we first list the response variable, and with the tilde symbol we indicate that we model it as a function of the two predictors, and we assume their effects to be additive. Finally, we tell R where to find the variables.

When we test for effects like this, we should not only look at p-values but consider three aspects: the direction of the effect, the size of the effect, and the statistical significance. In the regression output, the sign of the slope coefficients tells us which way the effect goes. Here we see that both predictors have negative slopes. This means that site-level FST decreased with increasing population size and connectivity, just as we expected. In this example, we are not really interested in the intercept, we pretty much ignore it.

Are the effects of population size and connectivity statistically significant? The p-value of the t-test for each slope coefficient tells us whether it is significantly different from zero. This is all about whether or not we can rule out chance. Both p-values are smaller than 0.05, hence we rule out chance and accept the alternative hypothesis that there is an effect. However, this does not prove that population size or connectivity caused the observed effect, because this is only an observational study, not a controlled experiment. And it also does not tell us directly whether population size and connectivity had a large effect on differentiation.

There are three ways we can look at effect size here. The first is the multiple Rsquared. It tells us how much variance in site-level FST was explained by the two predictors together: 46 percent, that's actually quite a lot. However, we typically use the adjusted Rsquared, which is a little lower, here 41 %, because it has a penalty for the number of predictors in the model.

If we want to know which variable was more important, population size or connectivity, we could compare their slope estimates but these depend on the units, and population size and connectivity are measured in different units. If we re-run the analysis with all variables standardized, including the response, then we get the beta coefficients, which we can interpret like correlation coefficients. A value of 1 would be a perfect positive correlation, and zero means no correlation. These are partial correlations, which means for example that the partial correlation between site-level FST and connectivity is calculated after accounting for the effect of population size. Therefore, the value depends on what other variables are in the model. By the way, the same is true for the p-values that we already looked at, they are calculated after accounting for all other variables in the model, and they may change if you add or drop a predictor variable.

The scientific notation is a bit confusing, but we see that the partial correlation for population size is 0.525, and 0.475 for connectivity. Population size thus has a slightly larger effect than connectivity, when considered in the same model. But how large is each effect, is it large enough to be biologically relevant? This may depend on the research question. However, Cohen came up with a rule of thumb for making effect sizes comparable between different data sets and analysis methods. As a rule of thumb, a correlation greater than 0.1 would be considered a small effect, a correlation above 0.3 a medium effect, and above 0.5 is a large effect. So here, population size would qualify for a large effect and connectivity for a medium effect. Again, this is just a rule of thumb. If we square the correlations, we get Rsquared, so now we can apply the same rule to the adjusted Rsquared, and taken together, the two predictors had a large effect on site-level FST.

So how much of this variance is explained by each predictor individually, and how much is shared between the two? We can answer this with variation partitioning. I won't go into details here but in this example, population size alone accounts for 26% of variance, connectivity for 20%, and there is actually no shared variance, the two predictors are pretty much independent of each other. That's a good sign, because then the slope estimate of one variable will not be much affected by the presence of the other variable in the model.

To summarize, we found that population size and connectivity both have a substantial and statistically significant effect on variation in site-level FST, and the effect was consistent with our expectation. The result confirms our hypothesis derived from metapopulation theory, but it does not establish a causal relationship because the data come from an observational study.

Scene 6: Checking Assumptions

However, we were a bit too quick here. Before we make such conclusions, we need to check whether the model is actually valid, or whether any statistical assumptions were violated in important ways. The assumptions of Ordinary Least Squares or OLS regression concern three aspects: the sampling design, the response variable, and the predictor variables.

When we do a hypothesis test, the hypothesis concerns the underlying statistical population, not the sample. The statistical population would be all sites that we potentially could have sampled. The assumption is that our sample is a simple random sample of these sites. Or for an experiment, that the experimental units were randomly assigned to treatments. Not all landscape genetic studies use a statistical random sample of sites, and this limits our ability to generalize beyond the actual sample.

A whole set of assumptions and conditions refers to the response variable, and we check these with residual analysis. Here we have the default figure produced by R, with four plots. The first one shows the residuals plotted against the fitted values. Regression assumes that the relationship between response and predictors is linear, and if that is the case, the points in this plot should form a horizontal band, and the smooth red line should be more or less horizontal, not curved. Here we see some curvature, with positive values on the left, negative in the middle, and again positive values on the right. We should follow up with plotting FST against each predictor variable separately.

A second assumption is that the errors are normally distributed, and we check it with a normal probability plot. The points should more or less fall on a straight line. Here they follow a line for the most part, but on the right side, the last three values seem off, they are higher than expected. There's always more variability at the ends, but this seems to indicate a systematic deviation.

Another assumption is that the variance is constant, and we check this with a plot of the absolute value of the residuals against the fitted values. This is almost the same plot as the first one, only now all values are positive. Again, we want to see a horizontal point cloud, and a more or less horizontal smooth red line. And again, there is some deviation, with an increase in variance for the largest fitted values on the right end of the plot.

In all three plots, it was the sites number 32 and 42 that were off. Are these somehow different from the rest? Another regression assumption is that the residuals are identically distributed, which would be violated if these two sites were somehow fundamentally different from the others. Should we treat them as outliers and potentially remove them? On one hand, this depends on whether we have a good biological justification for removing them. On the other hand, we want to know whether they have a big influence on our statistical result. The fourth plot shows us just this. On the y-axis, we have the residual, and as before, sites 32 and 42 have the largest residuals. A large residual will lower the R-squared of the model. On the x-axis, we have a measure of the leverage effect of each site. This means, how much the slope estimates would change if this site was omitted. The total influence of a site on the regression results is a combination of the two, and we can quantify this with Cook's distance. We could plot Cook's distance directly, but here it is added as contour lines. Any site with Cook's distance larger than 1 is typically considered an influential point.

In summary, there were multiple concerns that were related to sites 32 and 42, though neither site had a strong influence on the overall results. We should have a closer look to find out what is going on to decide whether a transformation would help stabilize the variance, or whether we may need to include an additional predictor variable, or whether we have convincing biological reason to exclude any site because it is affected by different processes than the other sites.

There is one more assumption concerning the response, and that is that the errors are independent. One way this could be violated is if we have spatial autocorrelation, where nearby sites have more similar residuals than distant ones. We'll learn next week how to check this.

A third set of assumptions is about the predictor variables. They should be measured without error, and they should not show multi-collinearity. What is this? In the strict sense, we should not have any redundant variables that can be fully replaced by any combination of the other predictor variables, because then the method won't work. In the wider sense, the predictor variables should not be strongly correlated among each other, because then the slope estimates would change a lot depending on which variables are included in the model. In statistical terms, this would increase the uncertainty of our slope estimates and possibly make our p-values unreliable. We can check for this with the Variance Inflation Factor. It is a function of how much of the variance of one predictor we can explain by the other predictors in the model. The interpretation is that the standard error of the slope estimate is inflated by the square root of the Variance inflation factor. So, if the variance inflation factor is 4, this would double the confidence interval of our slope estimate. There is no real consensus about where to draw the line. A variance inflation factor of 1 indicates no collinearity, a value up to five is often considered moderate collinearity, and most would consider a value above 10 problematic.

This stats review brings us to the end of the first video. A second video explains R graphics, especially how to use 'ggplot'.

VIDEO 2

Scene 1: Metapopulation Genetics

In this second video for Week 4, we will look at R graphics. First, we will create the same figure twice, once with basic R graphics and once with the ‘ggplot2’ package, and then we will plot residuals in space.

Scene 7: The R Roller Coaster

Working with R can be an emotional roller coaster, and that is especially true for R graphics. The learning curve is pretty steep, there is no graphics wizard or clickable menu but instead, you have to program every detail. And there are so many details and options and arguments that it seems impossible to remember them.

On the other hand, R gives you full control over print-quality figures, and because every detail is programmed, we can simply save the code and recreate the figure later. This comes in really handy when you need to tweak a figure to satisfy an editor and reviewer. And then, there are some advanced features that let me do cool things I never thought I could do.

My strategy in the uphill battle with R graphics is three-fold: first, understand how R thinks. Second, google it. If you know what you want to get done, chances are someone has already posted a blog with a worked example. Third, always save the code to reproduce your figures, and re-use the code next time you need to do something similar.

The hardest step may be the first: understand how R thinks. And for this, we’ll need a historical perspective again. Remember this timeline when R branched off from S? Base R graphics date back to early versions of S. Since then, there have been two graphics revolutions, each tied to a book. The first was ‘Visualizing Data’ by Cleveland, which introduced Trellis graphics. This approach is implemented with the ‘lattice’ package and is used for example by the ‘sp’ package for plotting spatial objects in space. The second book was ‘The Grammar of Graphics’ by Wilkinson, which has been implemented with the ‘ggplot2’ package by Hadley Wickham, who is taking a leading role in data analysis and visualization with R from his current position as chief scientist at RStudio.

This means that we have three different graphical paradigms, or philosophies, encoded in R functions. Each approach is highly flexible in itself, but they don’t mix and match.

Scene 8: Basic R Graphics

Here’s our example figure: a panel with two scatterplots of site-level FST against long-term population size NLT, on the left, and against connectivity C, on the right. A regression line is shown for each plot, and the two potential outliers that we identified already are marked with a red symbol and labeled with their pond ID.

Let’s start with the workflow first. How do we create a multi-panel plot like this, and how do we export it from R? To write your figure into a file, for example here a png graphics file, we call the ‘png’ function and specify the name of the output file. We also need to indicate the size of the canvas that we want, here 7 inches wide times 3.5 inches high, with a 12-point font size and a resolution of 300 dots per inch. Behind the scenes, ‘png’ opens a graphics device, and when we’re done with the figure, we need to close this device with the command ‘dev.off’. Only then will R actually write the file. PNG is great if you want to import the figure into other documents, like your manuscript or Powerpoint slides. For stand-alone files in print quality, I would use PDF. I would not recommend JPEG for R graphics because it uses a compression that loses information.

Before we create the two plots for the figure, we set a few parameters for the entire figure with the function ‘par’, which has a long list of optional arguments to tweak many details. Here, the argument ‘mfrow’ defines a multi-frame figure layout with one row and two columns, that means, two side-by-side plots. The argument ‘mar’ defines the margins of each plot. Each plot has four sides, and they are numbered clockwise starting at the bottom. The argument ‘mar’ takes a vector with four values, one for each side, and the numbers indicate how many lines of space should be reserved. The vector 4, 4, 1, and 1 here results in wide margins at the bottom and on the left, where all the labels go, and narrow margins at the top and on the right.

Now we’re all set to create the two plots and run the entire code to create a file with the figure. The function ‘plot’ is very versatile and creates different default plots depending on the data types of the variables. A convenient way to specify the variables is this formula notation, where we plot a response y, on the y-axis, as a function of a predictor x, on the x axis. We can use just the variable names as long as we provide a data argument that tells R where to find the variables, here a data frame ‘df’. This is useful because it then uses the variable names as default axis labels.

For each plot, we’ll need a few more lines of code to add the line, red symbols and pond ID labels. Here we have some pseudo-code for one figure. We already saw the formula and the data argument. Here, ‘ylim’ specifies that the y-axis should be limited to the range from 0 to 0.3, and ‘type = n’ tells R not to plot the data just yet. Then we use ‘xlab’ and ‘ylab’ to provide custom axis labels. Because I specified ‘type = n’, which means ‘none’, R will first create an empty plot just with the axes and axis labels but without the data points. Then we add the data points with the function ‘points’ , and ‘cex’ is the character expansion factor that determines the size of the symbols. I’m doing this in two steps here to make it more comparable to ‘ggplot’ on the next slide. To add the regression line, we fit a simple linear regression model with ‘lm’ and pass the fitted model to the function ‘abline’, which is named after the intercept a and slope b that define a regression line.

So far so good. Now we need to select a subset of the data, with just the two potential outliers, so that we can plot and label them separately. The row names were ‘32’ and ’42’, and here we create an indicator variable ‘a’ that is TRUE if the row name is an element of the vector with the values 32 and 42, and FALSE otherwise. Now we can add points again, this time only for the rows selected by the indicator ‘a’, and we plot them with a different symbol, or point character ‘pch’. The default symbol is an empty black circle, symbol number one. Here we choose symbol number 16, which is a filled circle, and we change the color to ‘red’.

To add the pond ID labels, we use the function ‘text’, again with the subset ‘a’ of the data frame ‘df’. Only, this function does not recognize the formula notation and does not take a data argument. As a workaround, we can use the function ‘with’ to tell R to execute the function ‘text’ with the variables in the object ‘df’. ‘SITE’ is the variable with the pond IDs, and we can specify the position where the labels should be drawn, that is, on which side of the symbol. The sides are again numbered 1 through 4, starting at the bottom, and 4 means that we plot the labels to the right of the symbols.

Scene 9: Grammar of Graphics: ‘ggplot2’

Here’s a similar two-panel figure created with the package ‘ggplot2’. It has a gridded background, a regression line with a 95% confidence envelope, and the two plots are labeled A and B.

How is the first plot constructed? One difference from the start is that we write each plot into an object, here Plot1. With the function ‘ggplot’, we create a plot using the grammar of graphics. Again we have a data argument, and we need to supply the two variables that define the x and y axes. This is done with a function ‘aes’ which stands for ‘aesthetics’, which I do not find intuitive at all.

Here's a schematic representation from the ggplot2 cheat sheet. ‘ggplot’ starts with a data frame, here with three variables F, M and A. We need to define the axes of the coordinate system, here F goes on the x-axis and A on the y-axis. Then we need to specify how each row is going to be represented. That’s done by defining a geometry: geom_point represents the data as points, geom_line as a line, and so on. If we want, we can vary the color of the point symbols according to some variable, here F, and the size according to another variable, here A, to create a bubble plot.

Let’s return to our scatterplot. We want to plot points, hence we add a layer with geom_point. Two details are interesting here. First, we literally use a ‘+’ sign to add a layer to the plot, and second, we don’t need to tell geom_point which points to plot, it will take that information from the aesthetics argument that we already provided. Now we can keep adding layers with the plus sign. For the regression line, we add a geometry layer ‘geom-smooth’, specify the line color as blue, and the linear shape of the line with the option ‘method = “lm”’. Then we call geom_point again with the subset ‘a’ to draw the red symbols for the two potential outliers.

As before, adding the pond ID labels is more complicated. For some reason, we have to define a new aesthetics argument with only the points that we want to label. Here, we indicate that the variable SITE should be used for labeling the two points. Because this is done within the aesthetics argument, it is going to be done separately for each data point, which means, each will get its own label. The remaining arguments define the size and position of the labels.

We write all of this code into the object Plot1, and we do the same for Plot2. Now we need to put them together into a multi-panel figure and export it as a PNG file. This is done with the function 'ggsave'. I can specifiy the file type by providing the file extension 'png', then I indicate again the size of the canvas, the units, and the resolution in dots-per-inch, dpi. In the argument 'plot', we use the function 'plot_grid' from the 'cowplot' package to tell R that we want to plot a grid of plots, here with the two plots Plot1 and Plot2, and that the plots should be labelled 'A' and 'B'.

Ouff, that was just a few lines of code for a pretty nice figure, but it really can be a bit confusing. Just keep in mind the steps, and how we've added the different layers to the base plot. Then everything just becomes variations on a theme.

There is a different way in which we could create a multi-panel figure, called faceting. We would use that to draw the same plot separately for different groups. Here for example we have the same scatterplot of site-level FST against population size, with once faceted plot for each cluster, and a fifth one for all the sites that do not belong to a cluster. The code to do this is very simple, we just add the function 'facet_grid' to the plot. We can even use two variables to define the facets, one for the rows and one for the columns.

Scene 10: Plotting Residuals in Space

Our last task here is to plot the residuals from the regression model in space. Here we write the fitted model into an object 'mod', and we can use the function 'attributes' to check what attributes are now stored in the object 'mod'. The residuals are stored in the second attribute, called residuals. Here we extract them and store them in their own object that we call 'Residuals' with capital R.

If we want to plot the residuals in geographic space, not just against another variable, then we need to deal with spatial coordinates and their projection. Here I'll show two strategies. The first one builds on the 'lattice' package. The package 'sp' that we've already used in Week 2 to store site data and spatial coordinates in a SpatialPointsDataFrame has functions for plotting variables in space. Here we have a bubble plot, where red indicates negative and blue positive residuals, and the size of each symbol indicates the size of the residual. We don't have a map of the island, but the ponds are drawn to scale, like on an invisible map. The code is surprisingly simple: the function 'bubble' of the 'sp' package takes a SpatialPointsDataFrame, dd.spatial, that defines the spatial locations, a variable 'zcol' to be visualized, here "Residuals", and optionally we can specify the colors, here red and blue.

With ‘ggplot’ we can go one step further and plot the points on an actual map of the island. The map is grabbed from the internet auto-magically, and the code is not that much longer. Here we use the function ‘qmplot’ for a quick map plot. We specify the data frame with the variables, including the coordinates in lat-long format, and we indicate the variables that hold longitude and latitude values. Internally, qmplot passes the information to the function ‘get_map’ that grabs a suitable map from the internet. Here we specify that the color should reflect the sign of the residuals and symbol size should reflect the absolute value of the residuals. To this map, we can add layers with ‘ggplot2’ functions. Here I used the geom_text layer from the previous slide to add the labels for the two potential outliers. Finally, we see that pond PTC is in an extremely isolated position on the tip of a peninsula, which could well explain why it showed such high genetic differentiation. For the other site, DESB, the map does not really explain what might be going on.

Scene 11: Map Types from Internet

Well, that was pretty impressive that we could get this map right from the internet. But, wouldn’t it be great if we could choose what type of map we want? Actually, we can do this by telling the ‘get_map’ function whether we want a terrain, satellite, toner or watercolor map, and whether it should grab it from Google or from Stamen. The top left map is a Google terrain map, which is very useful for showing the landscape context of our sampling sites. The top right is a grayscale map that is suitable for grayscale publications and makes a good background for plotting variables like our residuals.

Scene 1: Metapopulation Genetics

Now it’s your turn to apply this knowledge and produce cool figures and maps for your next project meeting or publication! Again, we’ll get some practice in the tutorial, and we’ll see figures and maps with ‘ggplot2’ in several of the worked examples. There is also an extra worked example on R graphics by Rodney Dyer, you can find it with the Add-in under Week 0.