Basic Population Genetics Analyses in R

Laboratory Exercise: Distributed Graduate Seminar

Rodney J. Dyer

Overview

In this lab we will be working with some genetic marker data to:

1. Learn about basic genetic data types in R as implemented in the gstudio package.
2. Compute summary statistics related to diversity for complete and subsets of a data set.
3. Estimate genetic distances for a set of populations and examine hypotheses of isolation by distance.
4. Measure genetic structure among populations in a nested model.

By the end of this exercise you should be able to effectively load in, manipulate, and analyze genetic data in R. This laboratory exercise will be using the R libraries `gstudio`, `ecodist`, and `pegas`, which is freely available from http://cran.r-project.org to download and install OR from within R you can just type:

```
> install.packages( c("gstudio","ecodist","pegas") )
```

Now that you have the libary on your machine, when you want to use it, you must type[1]:

```
> require(gstudio)
```

You will need to do this only once per session.

Genetic Data & R

The Locus Object

The reason for the `gstudio` library is that it defines some basic data types (like numeric, logical, etc) that describe genetic marker data. At the most fundamental level is the `Locus` object. A locus is collection of alleles but can be examined as a single entity like a floating point number or a factor, etc.

```
> loc1 <- Locus( c(120,122) )
> loc1
```

```
120:122
```

```
> loc2 <- Locus( c("A","T") )
> loc2
```

```
A:T
```

Note, that internally the alleles are translated into `character` objects. In all the functions dealing with alleles both `integer` and `character` arguments are accepted. There are several methods associated with the `Locus`, the main ones that you will be working with are shown below by example. See `help("Locus-class")` for a complete discussion.

Once we have a locus object in R, we can inquire about its genetic characteristics and access individual alleles if necessary (via the normal vector square bracket notation `[]` just like it was a regular vector.)

```
> loc3 <- Locus( c(122,122) )
> loc3
```

```
122:122
```

```
> is.heterozygote( loc3 )
```

[1]R has a *huge* number of libraries available but allows you to only keep in memory those that you need to use.

```
[1] FALSE
```

```
> loc3[2]
```

```
[1] "122"
```

```
> loc3[2] <- "124"
> is.heterozygote( loc3 )
```

```
[1] TRUE
```

```
> length( loc3 )
```

```
[1] 2
```

```
> summary( loc3 )
```

```
Class : Locus
Ploidy : 2
Aleleles : 122,124
```

The Population Object

Having a single locus is dandy, but where we want to be is to be working with populations of individuals, each of which has the potential to have several loci. In R, the `data.frame` is a general component that you keep your data in for analyses. The `gstudio` packages defines a `data.frame` for loci called a `Population`. A `Population` object can contain all the kinds of data that we typically have, such as:

1. Stata (popualtion names, regions, habitats, etc)
2. Spatial coordinates (latitude & longitude)
3. Other covariantes (slope, soil moisture, time since colonization, etc.)
4. Genotypes (haploid, diploid, dominant, etc.)

In this next example, I will define a `Population` object consisting of a set of populations, and environtmental variable measured at each individual, and two loci.

```
> strata <- c("Cabo","Cabo","Loreto","Loreto","Loreto")
> TPI <- c(Locus(c(1,2)),Locus(c(2,3)),Locus(c(2,2)),Locus(c(2,2)),Locus(c(1,3)))
> PGM <- c(Locus(c(4,4)),Locus(c(4,3)),Locus(c(4,4)),Locus(c(3,4)),Locus(c(3,3)))
> Env <- c(12,20,14,18,10)
> thePop <- Population( Pop=strata, Env=Env, TPI=TPI, PGM=PGM )
> thePop
```

```
     Pop Env TPI PGM
1   Cabo  12 1:2 4:4
2   Cabo  20 2:3 3:4
3 Loreto  14 2:2 4:4
4 Loreto  18 2:2 3:4
5 Loreto  10 1:3 3:3
```

```
> summary(thePop)
```

```
     Pop                 Env         TPI      PGM
 Length:5           Min.   :10.0   1:2:1   3:3:1
 Class :character   1st Qu.:12.0   1:3:1   3:4:2
 Mode  :character   Median :14.0   2:2:2   4:4:2
                    Mean   :14.8   2:3:1
                    3rd Qu.:18.0
                    Max.   :20.0
```

You can access elements of the population by either indexes or named columns and indexes:

```
> thePop[1,2]
```

```
[1] 12
```

```
> thePop$Pop[1:4]
```

```
[1] "Cabo"   "Cabo"   "Loreto" "Loreto"
```

```
> thePop[ thePop$Env>12,]
```

```
     Pop Env TPI PGM
1   Cabo  20 2:3 3:4
2 Loreto  14 2:2 4:4
3 Loreto  18 2:2 3:4
```

So when you take a partition of a `Population` object, it creates a new object with the subset of the data you used. Often we take partitions of data based upon strata (pop, region, etc) and for this, there is an easy function, `partition()` that will return a `list` object whose keys are the strata levels you request. For example, using the data above, I may want to partition the whole data set into populations and then estimate parameters (heterozygosity, allele frequencies, etc) on each stratum.

```
> pops <- partition(thePop, stratum="Pop")
> class(pops)
```

```
[1] "list"
```

```
> names(pops)
```

```
[1] "Cabo"   "Loreto"
```

```
> pops[[1]]
```

```
  Env TPI PGM
1  12 1:2 4:4
2  20 2:3 3:4
```

```
> pops["Loreto"]
```

```
$Loreto
  Env TPI PGM
1  14 2:2 4:4
2  18 2:2 3:4
3  10 1:3 3:3
```

```
> class( pops$Cabo )
```

```
[1] "Population"
attr(,"package")
[1] "gstudio"
```

Notice that you can access the `Population` objects in the list by either index number or population name. Having a list of `Population` objects is very helpful in R as you can extract lots of information from the list using the using the `lapply()` (list apply) function. For example, if you wanted to know the size of the population data sets across all strata you could type:

```
> lapply( pops, dim )
```

```
$Cabo
[1] 2 3

$Loreto
[1] 3 3
```

Where the `dim` is a function that returns the dimension (rows & columns) of the object. Here you can see that the `Cabo` stratum has 2 rows and 3 cols whereas the `Loreto` population has 3 rows and 2 columns. The `lapply` function can take more complicated functions than things like `dim`. In the following example, I show how to get the average of the `Env` variable by population using what is called an anonmymous function. Here I just make up a function that every element of the pops list will be passed.

```
> lapply( pops, function(x) mean( x$Env ) )
```

```
$Cabo
[1] 16

$Loreto
[1] 14
```

When R reads this, it says, hey I've got a list and I'm going to call every element of the list 'x' then I'm going to give it to function(x) and whatever it returns is what I'll print out (OK, R doesn't use subjective pronouns but you get the idea).

Reading Data From A File

R can read a wide variety of data formats. For this exercise, we will use a dataset that is included with the `gstudio` library itself so we will not have to worry too much about importing. What follows is a general overview, of how we can get data into R from a text file. In general, if you keep your data in a spreadsheet, you will export your spreadsheet as a CSV file and import that. For more info on how to do this, there is a much longer description of these methods in the vignette. Vignettes are short PDF documents that come with R packages that go into more depth about the functionality of the library. To see all the vignettes installed, at the R prompt type `browseVignettes()` and your browser will open and give you a list.

For this exercise, we will be using data that comes with `gstudio`. This data is from the bark beetle *Araptus attenuatus* that the Dyer laboratory has been working on in Baja California. To load and examine this data set type:

```
> data( araptus_attenuatus )
> summary( araptus_attenuatus )
```

```
   Species      Cluster         Pop        Individual       Lat
 CladeA: 75   CBP-C :150   32     : 19   101_10A:  1   Min.   :23.08
 CladeB: 36   NBP-C : 84   75     : 11   101_1A :  1   1st Qu.:24.59
 CladeC:252   SBP-C : 18   Const  : 11   101_2A :  1   Median :26.25
              SCBP-A: 75   12     : 10   101_3A :  1   Mean   :26.25
              SON-B : 36   153    : 10   101_4A :  1   3rd Qu.:27.53
                           157    : 10   101_5A :  1   Max.   :29.33
                           (Other):292   (Other):357
      Long            LTRS          WNT           EN           EF
 Min.   :-114.3   01:01:147   03:03  :108   01:01  :225   01:01:219
 1st Qu.:-113.0   01:02: 86   01:01  : 82   01:02  : 52   01:02: 52
 Median :-111.5   02:02:130   01:03  : 77   02:02  : 38   02:02: 90
 Mean   :-111.7               02:02  : 62   03:03  : 22   NA   :  2
 3rd Qu.:-110.5               NA     : 11   01:03  :  7
 Max.   :-109.1               03:04  :  8   03:04  :  6
                              (Other): 15   (Other): 13
    ZMP           AML           ATPS          MP20
 01:01: 46   08:08  : 51   05:05  :155   05:07  : 64
 01:02: 51   07:07  : 42   03:03  : 69   07:07  : 53
 02:02:233   07:08  : 42   09:09  : 66   18:18  : 52
 NA   : 33   04:04  : 41   02:02  : 30   05:05  : 48
             NA     : 23   07:09  : 14   05:06  : 22
             07:09  : 22   08:08  :  9   11:11  : 12
             (Other):142   (Other): 20   (Other):112
```

You can see that these data have categorical strata (species, cluster, pop, individual), spatial data (lat & long), and 8 codominant loci.

For the rest of the text, I am just going to use a subset of the data and allow you to answer some questions using the larger data set at the end. The subset I am going to use is the genotypes and populations from "Clade A" & "Clade B" (e.g., those in Baja California) and I'm going to call it 'pops' for brevity.

```
> pops <- araptus_attenuatus[ araptus_attenuatus$Species != "CladeB", ]
> summary(pops)
```

```
   Species      Cluster         Pop        Individual       Lat
 CladeA: 75   CBP-C :150   75     : 11   12_10A :  1   Min.   :23.08
 CladeC:252   NBP-C : 84   Const  : 11   12_1A  :  1   1st Qu.:24.59
              SBP-C : 18   12     : 10   12_2A  :  1   Median :26.02
              SCBP-A: 75   153    : 10   12_3A  :  1   Mean   :26.18
                           157    : 10   12_4A  :  1   3rd Qu.:27.53
                           160    : 10   12_5A  :  1   Max.   :29.33
                           (Other):265   (Other):321
      Long           LTRS          WNT            EN           EF
 Min.   :-114.3   01:01:146   03:03  :108   01:01  :218   01:01:196
 1st Qu.:-113.0   01:02: 69   01:03  : 76   01:02  : 52   01:02: 41
 Median :-111.7   02:02:112   02:02  : 62   02:02  : 38   02:02: 90
 Mean   :-111.9               01:01  : 53   01:03  :  5
 3rd Qu.:-110.7               03:04  :  8   01:04  :  4
 Max.   :-109.6               01:04  :  7   03:03  :  3
                              (Other): 13   (Other):  7
    ZMP           AML            ATPS          MP20
 01:01: 45   08:08  : 50   05:05  :155   05:07  :64
 01:02: 51   07:07  : 42   03:03  : 69   07:07  :53
 02:02:214   07:08  : 42   09:09  : 65   18:18  :52
 NA   : 17   04:04  : 41   07:09  : 14   05:05  :48
             07:09  : 22   08:08  :  9   05:06  :22
             08:09  : 22   03:06  :  3   06:06  :11
             (Other):108   (Other): 12   (Other):77
```

Allele Frequencies

One of the first things we do when examining population genetic structure is to look at allele frequencies. The `gstudio` package defines the `AlleleFrequency` data type that allows us to get individual frequencies, heterozygosities, etc.

You can get allele frequencies from a single locus

```
> freqs <- allele.frequencies( pops, loci="AML")
> freqs
```

```
$AML
Allele Frequencies:
  08 = 0.2664577
  09 = 0.1551724
  07 = 0.2507837
  10 = 0.01410658
  06 = 0.07680251
  11 = 0.001567398
  02 = 0.001567398
  13 = 0.001567398
  05 = 0.07523511
  01 = 0.001567398
  03 = 0.02037618
  04 = 0.1347962
```

a subset of loci

```
> freqs <- allele.frequencies( pops, loci=c("ATPS","LTRS"))
> freqs
```

```
$ATPS
Allele Frequencies:
  09 = 0.2262997
  07 = 0.03058104
  05 = 0.4816514
  10 = 0.004587156
  03 = 0.2155963
  02 = 0.006116208
  08 = 0.02752294
  01 = 0.003058104
  06 = 0.004587156

$LTRS
Allele Frequencies:
  01 = 0.5519878
  02 = 0.4480122
```

or from all loci

```
> freqs <- allele.frequencies( pops )
> names(freqs)

[1] "LTRS" "WNT"  "EN"   "EF"   "ZMP"  "AML"  "ATPS" "MP20"

> freqs$MP20

Allele Frequencies:
  07 = 0.2892308
  05 = 0.2969231
  15 = 0.001538462
  08 = 0.02769231
  06 = 0.08923077
  04 = 0.009230769
  18 = 0.1784615
  19 = 0.009230769
  17 = 0.04153846
  10 = 0.01384615
  11 = 0.04153846
  16 = 0.001538462
```

Once you have a frequency object, you can get both observed and expected heterozygosity (and inbreeding F if you like).

```
> ho( freqs$MP20 )

       ho
0.4246154

> he( freqs$MP20 )

      he
0.783787

> f <- 1- ho( freqs$MP20 ) / he( freqs$MP20 )
> f

       ho
0.4582515
```

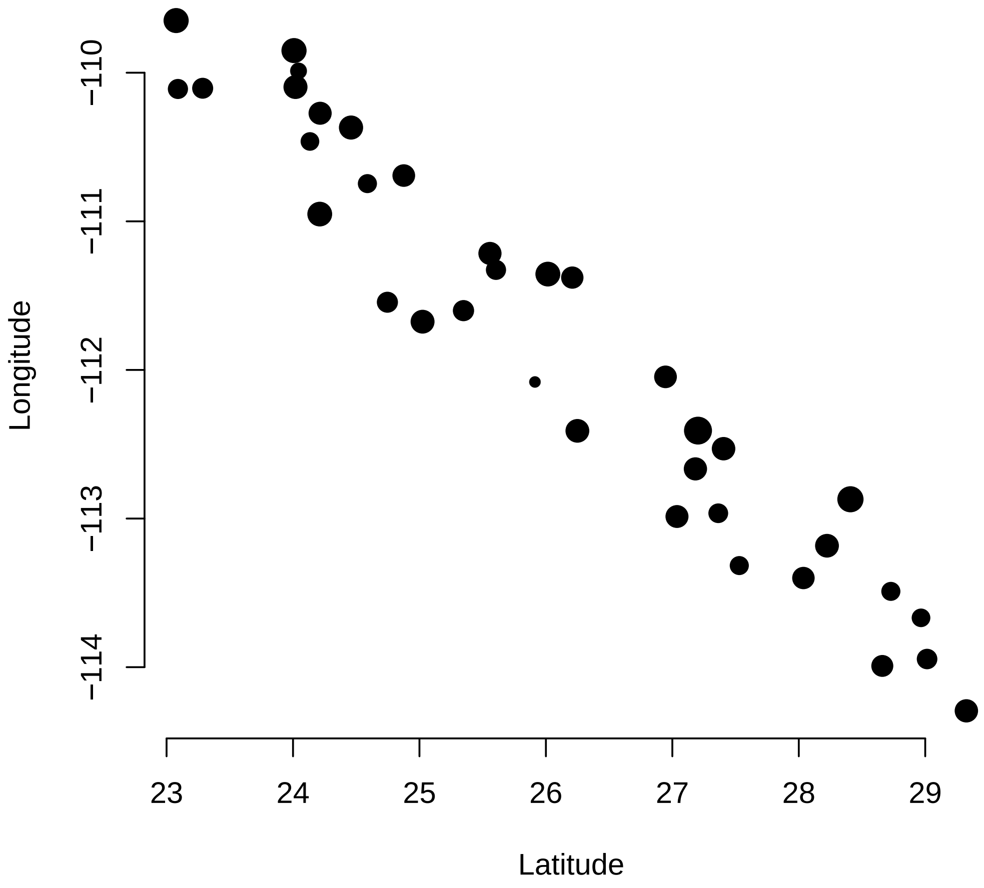
Warning: the `allele.frequencies()` function returns a list indexed by the name of the locus even if you only are asking for a single locus. We will see this behavior again when we estimate genetic distances.

So, now, it is possible to do something fun with the data. Lets plot the location of the populations again but this time lets make the symbol size scaled by H_e. This is the entire species' range we are examining and there are several kinds of core vs. periphery hypotheses that can be examined. What we will do is first use `lapply` as above to get H_e for all AML loci by population (and then take it out of a list using the `unlist`) function). Options I pass to the plot are `pch` which is the plot character (a filled circle), cex is the character expansion size (scaling of the symbol), and bty is the box type around the plot (I personally hate boxes around my plots). I scale the H_e values for visual differences because the size is proportional to heterozygosity, not the latitude/longitude.

```
> subpops <- partition( pops, stratum="Pop")
> he <- unlist(lapply( subpops, function(x) he(allele.frequencies(x,"AML")[[1]] ) ))
> he
```

```
   12.he    153.he    156.he    157.he    159.he    160.he    161.he    162.he 
0.6200000 0.5150000 0.4444444 0.4600000 0.5679012 0.7250000 0.6350000 0.5850000 
   163.he    164.he    165.he    166.he    168.he    169.he    171.he    173.he 
0.4600000 0.4850000 0.6250000 0.7900000 0.6150000 0.6050000 0.5950000 0.6350000 
   175.he    177.he     48.he     51.he     58.he     64.he     73.he     75.he 
0.5816327 0.6750000 0.1800000 0.5000000 0.6111111 0.5400000 0.6400000 0.5937500 
    77.he     84.he     88.he     89.he      9.he     93.he     98.he    Aqu.he 
0.5350000 0.4609375 0.6728395 0.6500000 0.4444444 0.6100000 0.3750000 0.6428571 
 Const.he   ESan.he    Mat.he    SFr.he 
0.6850000 0.5312500 0.5000000 0.6851852 
```

```
> scaled_he <- 2*he + .5
> lat <- unique(pops$Lat);
> lon <- unique(pops$Long)
> plot( lat, lon, xlab="Latitude", ylab="Longitude", pch=16, cex=scaled_he, bty="n")
```



Genetic Diversity

There are several measures of genetic diversity, many of which we can quickly estimate in R.

Polymorphic Loci: The number of polymorphic loci. We can go through each population in the data set and for each locus determine if there are more than one allele (tedious). Or we can make R do it with a little looping or cleaverness (much more exciting). First I am going to define a function that takes a list of `AlleleFrequency` objects (what you get from a call to `allele.frequencies`) and finds out how many loci have a length > 1 (e.g., they have more than one allele). It returns the value as a fraction of how many total loci there are.

```
> polymorphic.loci <- function(x) {
+         sum(lapply( x, length ) > 1) / sum(length(x))
+ }
```

Next, I can apply this to all the data.

```
> all.freqs <- lapply( subpops, allele.frequencies )
> P <- lapply( all.freqs, polymorphic.loci )
> unlist(P)
```

```
   12   153   156   157   159   160   161   162   163   164   165   166   168
0.750 0.625 0.625 0.625 0.875 0.875 0.625 1.000 0.750 1.000 0.875 1.000 0.750
  169   171   173   175   177    48    51    58    64    73    75    77    84
0.750 0.875 0.875 0.625 0.625 0.375 0.625 0.625 0.500 0.625 0.875 0.750 0.875
   88    89     9    93    98   Aqu Const  ESan   Mat   SFr
0.750 0.750 0.625 0.750 0.625 0.875 1.000 0.625 0.750 0.750
```

This shows one of the real strengths of R. What we've done here is made a completely new function and can use it as long as it is in memory or can save it to our own personal library of functions.

Allelic Diversity: Allelic diversity can be quantified as the number of alleles at a locus (A), the number of alleles at a locus above some predefined frequency (A_{95} for those whose frequencies exceed 5%), and the effective number of alleles (A_e). The `genetic.diversity` function allows you to estimate all of these parameters and does so using rarefaction (e.g., permutations to test diversity for standardized sample sizes, it may take a minute to do it on your computer).

```
> diversity.mp20 <- genetic.diversity( pops, stratum="Pop", mode="Ae",loci="MP20")
> diversity.mp20
```

```
Geneic Diversity:
  Estimator: Ae
  Stratum: Pop
  Loci: { MP20 }
  Locus = MP20
    12 Ae = 2.98507462686567 ; Rarefaction Ae = 2.576352595699
    153 Ae = 1.69491525423729 ; Rarefaction Ae = 1.70087039077751
    156 Ae = 2.18181818181818 ; Rarefaction Ae = 2.02766271348315
    157 Ae = 2.73972602739726 ; Rarefaction Ae = 2.34135280314847
    159 Ae = 2.41791044776119 ; Rarefaction Ae = 2.11275046969577
    160 Ae = 3.33333333333333 ; Rarefaction Ae = 2.78634896447074
    161 Ae = 4.16666666666667 ; Rarefaction Ae = 3.13368481814302
    162 Ae = 3.2258064516129 ; Rarefaction Ae = 2.62689419958082
    163 Ae = 2.17391304347826 ; Rarefaction Ae = 2.01553350560574
    164 Ae = 1.85185185185185 ; Rarefaction Ae = 1.78996445997478
    165 Ae = 3.17460317460317 ; Rarefaction Ae = 2.59848584057734
    166 Ae = 3.125 ; Rarefaction Ae = 2.72344186794823
    168 Ae = 2.89855072463768 ; Rarefaction Ae = 2.45144682032195
    169 Ae = 2.1978021978022 ; Rarefaction Ae = 2.0421096990206
    171 Ae = 2.1978021978022 ; Rarefaction Ae = 1.96317772751209
    173 Ae = 2.46913580246914 ; Rarefaction Ae = 2.19183418199586
    175 Ae = 1.96 ; Rarefaction Ae = 1.79530118353648
    177 Ae = 1.8018018018018 ; Rarefaction Ae = 1.72853823491594
    48 Ae = 1.6 ; Rarefaction Ae = 1.55902726255667
    51 Ae = 1.55555555555556 ; Rarefaction Ae = 1.56073680527757
    58 Ae = 1.90588235294118 ; Rarefaction Ae = 1.83654635084976
    64 Ae = 1 ; Rarefaction Ae = 1
    73 Ae = 2.17391304347826 ; Rarefaction Ae = 2.07897901095011
    75 Ae = 2.3047619047619 ; Rarefaction Ae = 2.1009077435634
    77 Ae = 2.10526315789474 ; Rarefaction Ae = 2.03445786303715
```

```
        84 Ae = 1.90588235294118 ; Rarefaction Ae = 1.80051656121832
        88 Ae = 1.6 ; Rarefaction Ae = 1.55172348819408
        89 Ae = 2.06185567010309 ; Rarefaction Ae = 1.89543970175549
        9 Ae = 1.11724137931034 ; Rarefaction Ae = 1.11980686568922
        93 Ae = 2.98507462686567 ; Rarefaction Ae = 2.54567897347395
        98 Ae = 1.68421052631579 ; Rarefaction Ae = 1.65878854006821
        Aqu Ae = 2.8 ; Rarefaction Ae = 2.4462072725437
        Const Ae = 2.37254901960784 ; Rarefaction Ae = 2.1865324214963
        ESan Ae = 2.24561403508772 ; Rarefaction Ae = 2.12420737288986
        Mat Ae = 1.28 ; Rarefaction Ae = 1.29617853147265
        SFr Ae = 2.18918918918919 ; Rarefaction Ae = 2.00934647747169
```

Heterozygosities: In the previous section we went into heterozygosity estimation by populations so we will probably not need to cover it again here.

Genetic Distance

There are many different kinds of genetic distance and `gstudio` hides them within the function `genetic.distance`. When estimating distance, you must pass a population, the name of the stratum on which to partition, and the mode of the distance calculation. See `?genetic.distance` for more information. At present the following distance types are available:

Indiviudal Distances: These are distances measured among individuals. The resulting matrices will be NxN in size.

- *Jaccard*
- *Bray*
- *AMOVA*

Stratum Distances: These are estimated among strata resulting in a matrix of size KxK

- Euclidean
- Cavalli-Sforza
- Nei
- cGD

There is a corresponding distance metric that can be estimated from coordinates using `stratum.distance` that will return the "great circle distance" (or euclidean) from a set of strata. What we'll do is measure Nei's distance among populations (need a $-1 * log(nei)$ for standard Nei's distance) in the data set and then plot that against physical distance (IBD). When we use this, we'll test for a correlation using a Mantel test from the `ecodist` library.

```
> phys <- stratum.distance(pops,stratum="Pop",lat="Lat", lon="Long")
> nei <- genetic.distance( pops, stratum="Pop", mode="Nei")[[1]]
> require(ecodist)
> mantel( as.dist(phys) ~ as.dist(nei) )
```

```
   mantelr      pval1      pval2      pval3  llim.2.5% ulim.97.5%
-0.6232364  1.0000000  0.0010000  0.0010000 -0.6611961 -0.5846042
```

For some distance metrics, there are alternative ways to accumulate distances across loci. As such, I have left the individual loci separate and let you decide how to combine them. Here is an example of this using a `for`-loop adding all the single locus values.

```
> cav <- genetic.distance(pops, stratum="Pop", mode="Cavalli")
> names(cav)
```

```
[1] "LTRS" "WNT"  "EN"   "EF"   "ZMP"  "AML"  "ATPS" "MP20"
```

```
> cav.all.loci <- matrix(0,nrow=36,ncol=36)
> for( locus in names(cav) )
+         cav.all.loci <- cav.all.loci + cav[[locus]]
> mantel( as.dist(cav.all.loci) ~ as.dist(phys) )
```

```
  mantelr      pval1      pval2      pval3  llim.2.5% ulim.97.5%
0.6333091  0.0010000  1.0000000  0.0010000  0.5905275  0.7010702
```

Genetic Structure

Genetic structure is a measure of among-strata configuration. In the lecture we examined F_{ST}, G_{ST}, & Θ as population parameters, G'_{ST} & D_{est} as population parameters standardized for highly diverse loci, and Φ_{ST} as a multilocus statistical measure of differentiation. Population structure parameters are fundamental tools for population genetics and have been perhaps, the most poorly understood and misused as well.

These structure parameters are estimated using the function `genetic.structure` and requires a `Population` object, a `stratum`, the `loci` you want to estimate parameters from, and a `mode` (the parameter you want). If you leave off the `loci` parameter, all loci will be used. There is also an optional parameter, `num.perm` that is used to test significance. In what follows, we will examine the following parameters as a demonstration of how to estiamte these parameters:

G_{ST}: This parameter is estimated from the differences in observed and expected heterozygosity.

```
> gst <- genetic.structure( pops, stratum="Pop", loci="EN", mode="Gst", num.perm=0)
> gst
```

```
Geneic Structure Analysis:
  Estimator: Gst
  Stratum: Pop
  Loci: { EN }
   - EN ;  Gst = 0.345786963051191
```

G'_{ST} **&** D_{Est}: These parameters provide corrections that are caused by loci with high allelic diversity.

```
> gst.prime <- genetic.structure(pops, stratum="Pop", loci="EN", mode="Gst.prime", num.perm=0 )
> gst.prime
```

```
Geneic Structure Analysis:
  Estimator: Gst.prime
  Stratum: Pop
  Loci: { EN }
   - EN ;  Gst.prime = 0.459618108931204
```

```
> dest <- genetic.structure(pops, stratum="Pop",loci="EN",mode="Dest", num.perm=0)
> dest
```

```
Geneic Structure Analysis:
  Estimator: Dest
  Stratum: Pop
  Loci: { EN }
   - EN ;  Dest = 0.164464814867075
```

Pairwise Structure

There are times when we may be intersted in estimating pairwise structure parameters. This can be done by invoding the optional `pairwise` flag for `genetic.structure`. Here is an example using the populations in Sonora (e.g., the non-peninsular populations).

```
> sonora <- araptus_attenuatus[araptus_attenuatus$Species=="CladeB",]
> genetic.structure(sonora,"Pop",loci="EN",mode="Gst.prime", pairwise=TRUE)
```

```
          101        32       102
101 0.0000000 0.4136727 0.3661894
32  0.4136727 0.0000000 0.1245850
102 0.3661894 0.1245850 0.0000000
```

AMOVA

The AMOVA is a statistical decomposition of genetic variance into two additive components, the within stratum component (σ^2_w) and the among strata component (σ^2_A). AMOVA produces a statistic, Φ_{ST} which is defined as:

$$\Phi_{ST} = \frac{\sigma_A^2}{\sigma_A^2 + \sigma_W^2}$$

using a distance matric approach. Here we will use the smaller sonora data set to decompose genetic structure and estimate this parameter.

```
> require(pegas)
> D <- genetic.distance( sonora, mode="AMOVA")[[1]]
> D <- as.dist(D)
> Pops <- as.factor( sonora$Pop )
> fit.amova <- amova(D ~ Pops)
> summary(fit.amova)
```

```
        Length Class      Mode
tab     3      data.frame list
varcoef 1      -none-     numeric
varcomp 2      data.frame list
call    2      -none-     call
```

```
> fit.amova
```

```
        Analysis of Molecular Variance

Call: amova(formula = D ~ Pops)

            SSD       MSD df
Pops   445.7229 222.86146  2
Error  919.2489  27.85603 33
Total 1364.9719  38.99920 35

Variance components:
      sigma2 P.value
Pops  17.773       0
Error 27.856

Variance coefficients:
       a
10.97222
```

Now unfortunately, the parameter Φ_{ST} is not directly produced by the output (don't know why but that is the way the author of the pegas library wrote it). However, it is easily calculated as:

```
> sigmaA <- fit.amova$varcomp[1,1]
> sigmaW <- fit.amova$varcomp[2,1]
> Phi <- sigmaA / (sigmaA+sigmaW)
> Phi
```

```
[1] 0.3895061
```

Exercises

The following exercises will allow you to test out the basic skills you learned in this laboratory.

```
> require(gstudio)
> data(araptus_attenuatus)
> data <- araptus_attenuatus[ araptus_attenuatus$Species=="CladeC",]
> counts <- table(data$Pop)
> counts
```

```
   12   153   157   159   160   161   162   163   164   165   166   168   169
   10    10     2     9    10    10    10     7     8    10     8    10    10
  171   173   175   177    51    58    64    73    75    77    84    88    89
   10    10     7    10     7     9     5     2     1     9     9    10    10
    9    93    98   Aqu Const  ESan   Mat   SFr
    9    10     1     4     3     2     1     9
```

If we look at this data, we can see we have a variable number of samples per population. In fact, for this Clade, there are several species with small sample sizes (as it turns out this is because what we thought was one species is actually two separate species in sympatry). So lets go through the data and remove those populations with fewer than 5 samples. If you look at the variable `counts` it is a numeric vector whose names are the population names. From this, we can find the population names whose counts are greater than 5

```
> keepers <- names(counts[ counts> 5 ])
> keepers
```

```
 [1] "12"  "153" "159" "160" "161" "162" "163" "164" "165" "166" "168" "169"
[13] "171" "173" "175" "177" "51"  "58"  "77"  "84"  "88"  "89"  "9"   "93"
[25] "SFr"
```

And then only use the data from those populations using the `%in%` operator.

```
> data <- data[ data$Pop %in% keepers, ]
> table(data$Pop)
```

```
 12 153 159 160 161 162 163 164 165 166 168 169 171 173 175 177  51  58  77  84
 10  10   9  10  10  10   7   8  10   8  10  10  10  10   7  10   7   9   9   9
 88  89   9  93 SFr
 10  10   9  10   9
```

This is pretty cool stuff because you can easily envision how easy it is to work with various subsets of your data set. OK, now on to some questions.

1. Can you rank these populations in terms of genetic diversity? What metric did you choose and why?
2. In the Clade C data, is there any indication of changes in expected heterozygosity as a function of either latitude or longitude? You can use the `cor.test()` function to test for significance.
3. In addition to strata-level genetic distances, there are also several individual-level genetic distance measures available. How correlated are the individual genetic distances from methods such as "AMOVA" and "Jaccard"? You may want to use the `mantel` function from the `ecodist` library as we did for population-level distances. Also, since the Jaccard distance is a single-locus estimates, you can either combine them across loci for a multilocus estimate or look at the loci individually.
4. I didn't use Bray-Curtis in the previous question because there are some missing data. Can you think of a way to handle missing data using this metric so that a comparison can be made?
5. Of the single-locus measures of genetic structure, which one would you use to estimate among-population structure and why? Is there a lot of structure in these data or a little?