Population Genetic Analysis of Mitochondrial DNA

Eric Crandall

Contents

Setup

Some lesson materials from Bruce Cochrane

```
library(knitr)
library(rentrez)
library(pegas)
library(ape)
library(tidyverse)
library(geomedb)
library(strataG)

# Use setwd() to move R to an appropriate directory on your conputer for this lesson.
```

DNA Sequence Variation and the Special Case of the Mitochondrial Genome

The most basic level of genetic variation is, of course, that of DNA sequence.

Organellar DNA has a rich history in population genetics. In the late 1970's, John Avise, then starting his career, proposed that animal mitochondrial DNA has some unique properties that make it suitable for use in population genetic studies. In particular

1. It is genetically haploid.
2. It does not recombine
3. It is uniparentally (usually maternally) inherited.
4. It is highly variable probably due to a high mutation rate owing to the fact that mitochondria are an oxidizing environment.
5. Perhaps most importantly, because it is haploid and maternally inherited (so 1/2 as many copies in 1/2 as many individuals), it has an effective size that is 1/4 that of nuclear genomes. This means that it experiences stronger genetic drift, and will be a leading indicator of population structure.

Fast forward to the present, and if you do a keyword search on "mitochondrial" in NCBI popset database it returns over 15,000 hits - in fact they far and away the most frequent class of sequence sets in that database. Note, however, that they are by and large restricted to animals - in plants, chloroplast DNA has been used successfully for similar purposes, however its lower level of variation reduces its utility somewhat.

While there is much we could say about the structure of mtDNA and its applicability to evolutionary questions, two features should be mentioned:

1. The *CO1* gene, or Cytochrome Oxidase subunit 1, a relatively conserved sequence, has been widely employed as a "DNA Barcode" to identify samples at the species level.
2. The "D loop", the origin of replication, is a very A/T-rich region that is highly variable among individuals. This is because it is a non-coding sequence that is not under purifying selection. This is an excellent sequence to use for population structure studies.

The intent here is to introduce mtDNA manipulation, so that you can consider it for further explorations. We will start by showing how we can use it to construct a network of haplotypes and ask whether it reveals meaningful relations among individuals in a population.

An example with dogs

The data we are going to look at are from popset 322367799, a collection of control region sequences from various dog breeds. To work with the popset database, we first have to query it, then fetch the data in very raw text format, then write the data to a text file, and read it back into R in FASTA format.

```
dat.rnt <-entrez_search(db="popset",term="322367799")
dat.raw <-entrez_fetch("popset",id=dat.rnt$ids,rettype="fasta")
write(dat.raw,file="dog.fasta")
dog.dat <-read.FASTA("dog.fasta")
dog.dat
```

```
## 17 DNA sequences in binary format stored in a list.
##
## Mean sequence length: 1270.588
##    Shortest sequence: 1248
##     Longest sequence: 1272
##
## Labels:
## HQ845266.1 Canis lupus familiaris isolate K_30 breed Chihuah...
## HQ845267.1 Canis lupus familiaris isolate K_31 breed Shih Tz...
## HQ845268.1 Canis lupus familiaris isolate K_32 breed Jack Ru...
## HQ845269.1 Canis lupus familiaris isolate K_33 breed Hovawar...
## HQ845270.1 Canis lupus familiaris isolate K_34 breed Malthez...
## HQ845271.1 Canis lupus familiaris isolate K_35 breed Poodle ...
## ...
```

```
##
## Base composition:
##     a     c     g     t
## 0.285 0.260 0.137 0.318
## (Total: 21.6 kb)
```

We need to edit the names of each sequence to something meaningful, in this case the breed. We can use the following quick trick for this case (for others, another parsing method might have to be applied)

```r
nms <- names(dog.dat)
nms<- sapply(nms, function(x) strsplit(x," ")[[1]][8])
names(dog.dat) <- nms
dog.dat
```

```
## 17 DNA sequences in binary format stored in a list.
##
## Mean sequence length: 1270.588
##    Shortest sequence: 1248
##     Longest sequence: 1272
##
## Labels:
## Chihuahua
## Shih
## Jack
## Hovawart
## Malthezer
## Poodle
## ...
##
## Base composition:
##     a     c     g     t
## 0.285 0.260 0.137 0.318
## (Total: 21.6 kb)
```

```r
#image.DNAbin(dog.dat)
```

Aligning the data

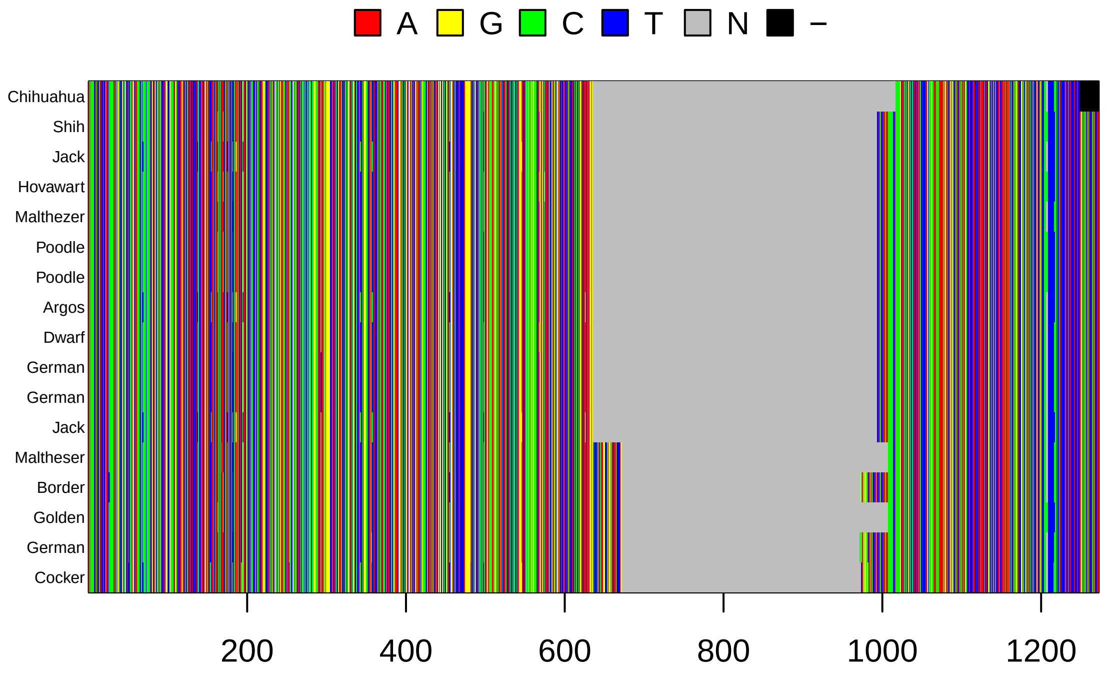
We can't get an image of the data unless all sequences are of the same length. What is going on?

We want to be sure they are properly aligned. To do so, we will use the add-on program `muscle` (which must be downloaded independently installed so that it can be called from R).

```r
dog.aln <-muscle(dog.dat)

#or alternatively, read it in from here
#dog.aln<-read.dna("https://ericcrandall.github.io/BIO444/Data/aligned_dogs.fasta",
#format = "fasta", as.matrix=T)
```

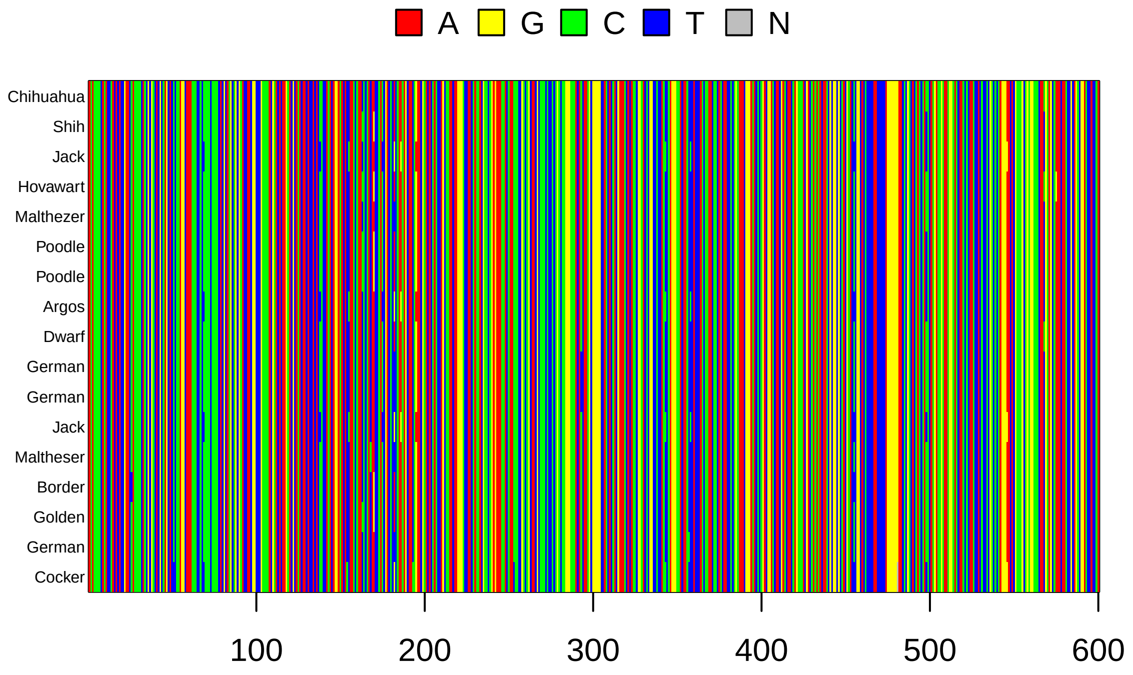
And with that, we can visualize the alignment:

```
image.DNAbin(dog.aln,cex.lab=.5)
```



We see that there is a huge gap in the middle of it. Here's where one of the great advantages of the matrix format comes in - we can readily select the first 600 nucleotides, which are free of gaps, and use them for our subsequent analysis:

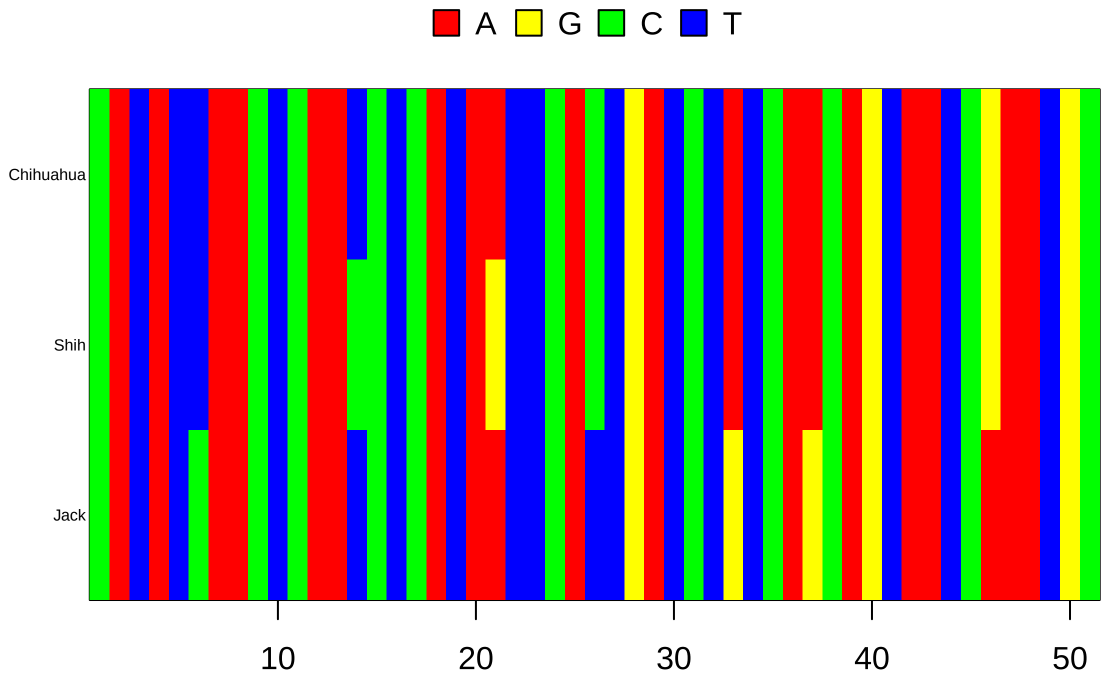
```
dog.aln.sub <-dog.aln[,1:600]
image.DNAbin(dog.aln.sub,cex.lab=.5)
```



And that looks better. Note that the names of the sequences were edited prior to loading the data as well - full original designations can be found in the popset link.

We can zoom in for a closer look

```
image.DNAbin(dog.aln.sub[1:3,150:200],cex.lab=.5)
```



And with this, we can, at least qualitatively, address our question. The fact that most bases are identical in all 17 sequences tells us that the sequence alignment looks good; the fact that there are some exceptions to this says that there is in fact *genetic variation*. Genetic variation provides the breadcrumbs that allow us to study the genetics of populations.

Measuring Genetic Diversity

But how much variation? There are three measures that are used extensively for this:

Number of segregating sites Since we are interested in variation, our attention should be focused on those sites that are in fact *polymorphic*, meaning that they have more than one type of nucleotide (they are therefore Single Nucleotide Polymorphisms (SNPs), or segregating sites). The ape package provides a function to give us those sites:

```
dog.sites <-seg.sites(dog.aln.sub)
dog.sites
```

```
##  [1]  26  51  69 138 154 155 163 168 170 175 182 186 193 195 253 293 343 357 358
## [20] 455 498 546 568 575
```

And we can count them rather easily

```
dog.ss <-length(dog.sites)
dog.ss
```

```
## [1] 24
```

And we see there are 24 out of the total of 600 positions that show some degree of variation. These are what we need to focus on - from a population perspective the remaining 585 sites tell us nothing.

Average pairwise differences OK, so we have 24 segregating sites. But there is another question we need to ask - on average, how many differences are there between two sequences?

Let's choose a couple of sequences at random and see how many differences there are between them.

```
seqs <-sample(1:17,2)
seqs
```

```
## [1]  3 11
```

Now let's see how many segregating sites there are.

```
ss <- seg.sites(dog.aln.sub[seqs,])
ss
```

```
##  [1]  69 138 155 175 182 186 195 293 343 358 455 498 546
```

```
ss_tot <- length(seg.sites(dog.aln.sub[seqs,]))
ss_tot
```

```
## [1] 13
```

And we see there are 69, 138, 155, 175, 182, 186, 195, 293, 343, 358, 455, 498, 546 segregating sites between these two randomly chosen sequences. If we reran the above block of code, we could select two other sequences, and we'd get another number. In fact, there are n(n-1)/2 such comparisons; the average of them is our measure of diversity.

$$\pi = \sum_{ij} x_i x_j \pi_{ij}$$

Where x_i and x_j are the respective frequencies of the ith and jth sequences in the sample and π_{ij} is the number of nucleotide differences between these two sequences.

Again, there is an R function (in the package pegas) that gets at this

```
dog.ndiv <- nuc.div(dog.aln.sub)
dog.ndiv
```

```
## [1] 0.01161249
```

But this is an important point to note. What we see here is the average diversity *per nucleotide*. In other words, if we were to pick a random position from two random sequences, this would be the probability that they would be different bases. We will use this occasionally, but for now, something that will be much more useful will be π, or the average number of differences *per sequence*. We can get this simply by multiplying the diversity number by the length of the sequence (600 in this case).

```
dog.pi <- ncol(dog.aln.sub) * dog.ndiv
dog.pi
```

```
## [1] 6.967495
```

This number tells us the average number of differences between any two sequences.

Haplotype Diversity A final question we can ask is: how unique is any given "haplotype" in this dataset? A haplotype (think "haploid genotype") here is simply the string of bases that are found on one chromosome, but if we had data from multiple loci, it would still be all of the bases found on one chromosome.

```
dog.hap <-haplotype(dog.aln.sub)
dog.hap
```

```
##
## Haplotypes extracted from: dog.aln.sub
##
##     Number of haplotypes: 12
##          Sequence length: 600
##
## Haplotype labels and frequencies:
##
##    I   II  III   IV    V   VI  VII VIII   IX    X   XI  XII
##    2    3    2    1    1    1    2    1    1    1    1    1
```

So among the 17 sequences in the data set, there are 12 different haplotypes. Now we can calculate haplotype diversity as:

$$h = \frac{N}{N-1}(1 - \sum_{i=1}^{n} X_i)$$

Where X_i is the relative haplotype frequency of each haplotype in the sample, and N is sample size. Haplotype diversity gives us the probability that two randomly chosen sequences in the sample will be different.

```
dog.hd <- hap.div(dog.aln.sub)

dog.hd
```

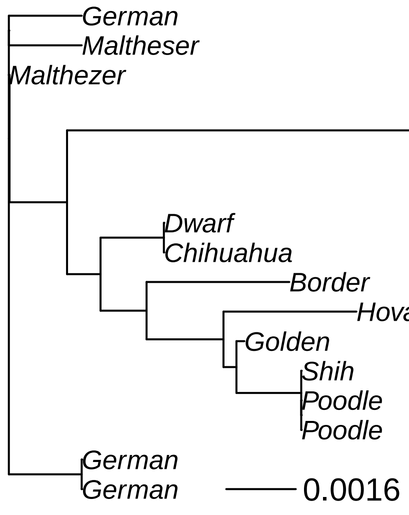
```
## [1] 0.9558824
```

Visualizing Sequence Variation

The color matrices of the data we have used so far are interesting, but not terribly pretty or useful. We can now better visualize the sequence data in two ways.

Visualization 1. A Phylogenetic Tree One of the most familiar ways to visualize relationships among aligned sequences is as a phylogenetic tree. In this case, what we will do is to measure the number of nucleotide differences between pairs of breeds as a measure of their genetic divergence and use the resulting matrix to generate a neighbor-joining tree. Note that for this we are treating all differences equally and are not taking the position of the differences into account.

```
dog.dist <-dist.dna(dog.aln.sub) #calculate the distance matrix
plot(nj(dog.dist),type="phylo",cex=.8) # plot a "neighbor joining" tree
add.scale.bar(x= 0.005, y = 1, length = 0.0016)
```

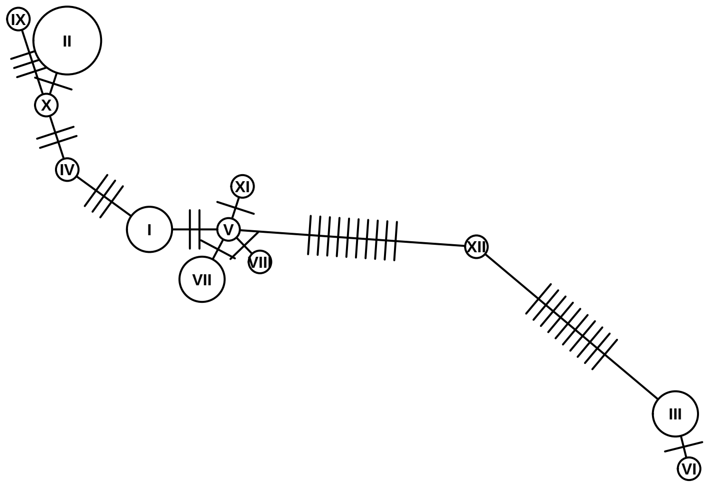


The scale bar indicates the branch length associated with 1 mutational change between sequences (p = 1/600)

But in fact, while useful, tree visualization like this is not entirely appropriate for population data, as it implies ancestor-descendant relationships that may or may not exist. In fact, we should be thinking of this as a network of haplotypes, which differ by one or more bases.

Visualization 2. A Network Another way of thinking about this is asking, given two haplotypes, how many substitutions would be required to go from one to the other. We would then extend that logic to create a network of all 12 haplotypes that requires the fewest steps. The code is a bit tricky, but it and the results are shown below:

```
net <-haploNet(dog.hap)
fq <-attr(net,"freq")
plot(net,size=fq,threshold=0,cex=.5)
```



The size of the circles indicate the number of sequences for each haplotype (for example, the smallest circles like X indicate a haplotype that occurred only once in the sample, while the largest on - II - actually had three). Note also that breeds are not shown - to do so would be a graphical nightmare. Instead, we can do a little bit of munging and generate a table that provides that information

```
sample.no <-attr(dog.hap,"index")


breed.by.haplo <-sapply(sample.no, function(x) rownames(dog.aln.sub)[x])
sorted <-sapply(breed.by.haplo,paste,collapse=" ")
haplo <-as.roman(1:12)
kable(data.frame(Haplotype=as.character(haplo),Breed=sorted))
```

Haplotype	Breed
I	Chihuahua Dwarf
II	Shih Poodle Poodle
III	Jack Jack
IV	Hovawart
V	Malthezer
VI	Argos
VII	German German
VIII	Maltheser
IX	Border
X	Golden
XI	German
XII	Cocker

So looking at these results, we can't say that we've learned a lot about these dog breeds. But after all, our sample size, both in terms of haplotypes and sequenced bases, is quite small. Nevertheless, it illustrates the potential of the approach.

GEOME Data

Let' try this with GEOME data!

Let's do some analysis on the ox-tongue Nerite snail *Nerita albicilla*. I've written a couple of papers on the population genetics of this snail. They are common algivores on coral reefs of the Indo-Pacific, and can typically be found at night on the underside of corals and boulders in shallow water. They can be identified by the little denticles on the inner lip of their shell.

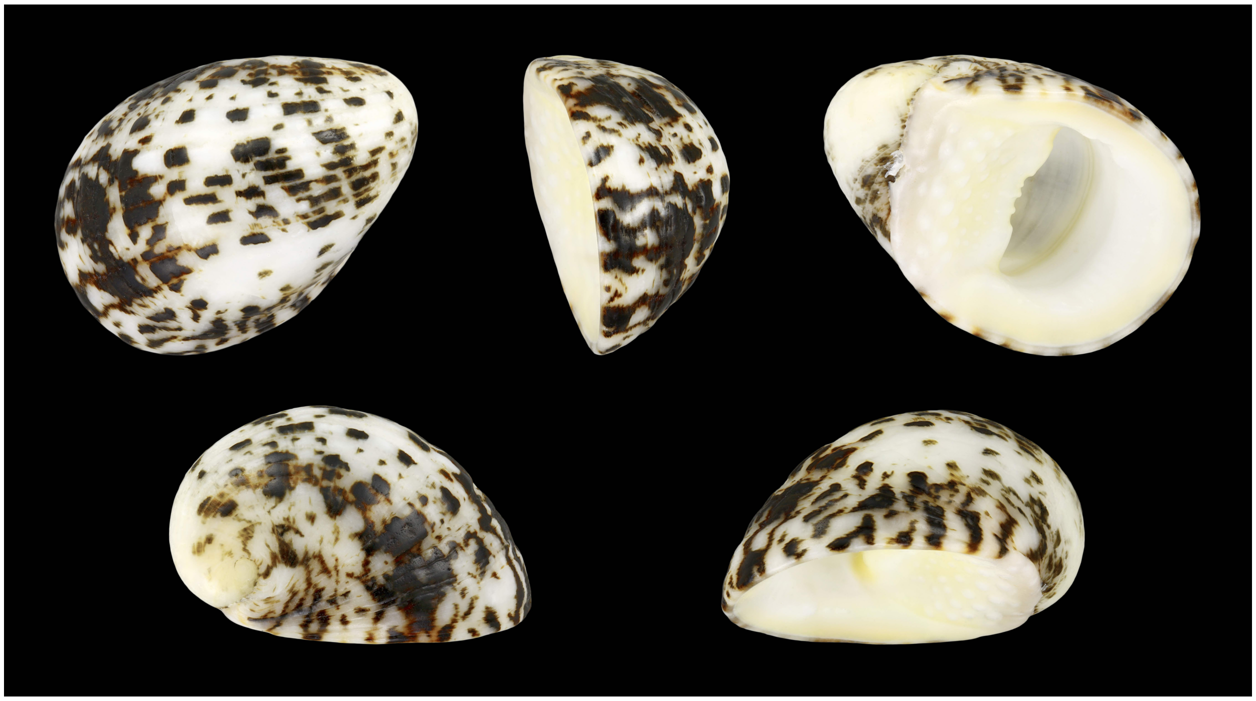


Figure 1: *Nerita albiclla*

Metadata

Let's download some metadata first. Metadata are data that describe other data. Here, the data are the genetic data and the metadata are data that describe each genetic sample: the taxonomy, the location sampled, the person who sampled it, the grant that payed for it, etc.

Relational databases aim to store data using what is called first normal form. We won't go into the entire meaning of this, but one of its major goals is to reduce redundant information.

If I visit a reef (lets say the island of Alam Kotok in Indonesia) and sample 20 *Nerita albicilla* from that reef, that is a sampling **event**. All of those 20 samples will have the same latitude, longitude, country, habitat, etc. So if I create a database of my samples, there is no need to repeat all this information for every single sample. Instead, it makes more sense to create one table that describes sampling **events**, and link it to a "daughter" table of **samples** that came from those events.

The link takes the form of a column of numbers (and/or letters) that uniquely identify each event. We will call this column the **eventID.** In the event table, the **eventID** is called the **primary key**. Now, instead of including all the event information (lat, long, country, etc.) when describing each sample, all I have to do is include a column in the **sample** table that is also called **eventID.** When it is found in the **sample** table, **eventID** is called a **foreign key**. If I use the **eventID** that I generated for Alam Kotok in the **Event** table, then all of the event metadata is effectively linked to each sample.

```
# First, let's query the metadata for all the Nerita albicilla samples
neralb_meta <- queryMetadata(
  entity = "Sample",
  query = "genus = Nerita AND specificEpithet = albicilla",
  select="Event")
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
# There are two tables within the linckia_meta object
#names(neralb_meta)

# one describes each sampling event (where, how, who)
#View(neralb_meta$Event)
# the other describes each sample (what was sampled)
#View(neralb_meta$Sample)
```

If we view the number of events, we find it to be much smaller than the number of samples. This makes sense, since multiple samples are taken at each event.

Let's rerun the query to just include *Nerita albicilla* from the Coral Triangle, an area of the Indo-Pacific that has the highest marine biodiversity on Earth, just like Amazonia is for terrestrial species. We will delimit it by decimalLongitude and decimalLatitude.

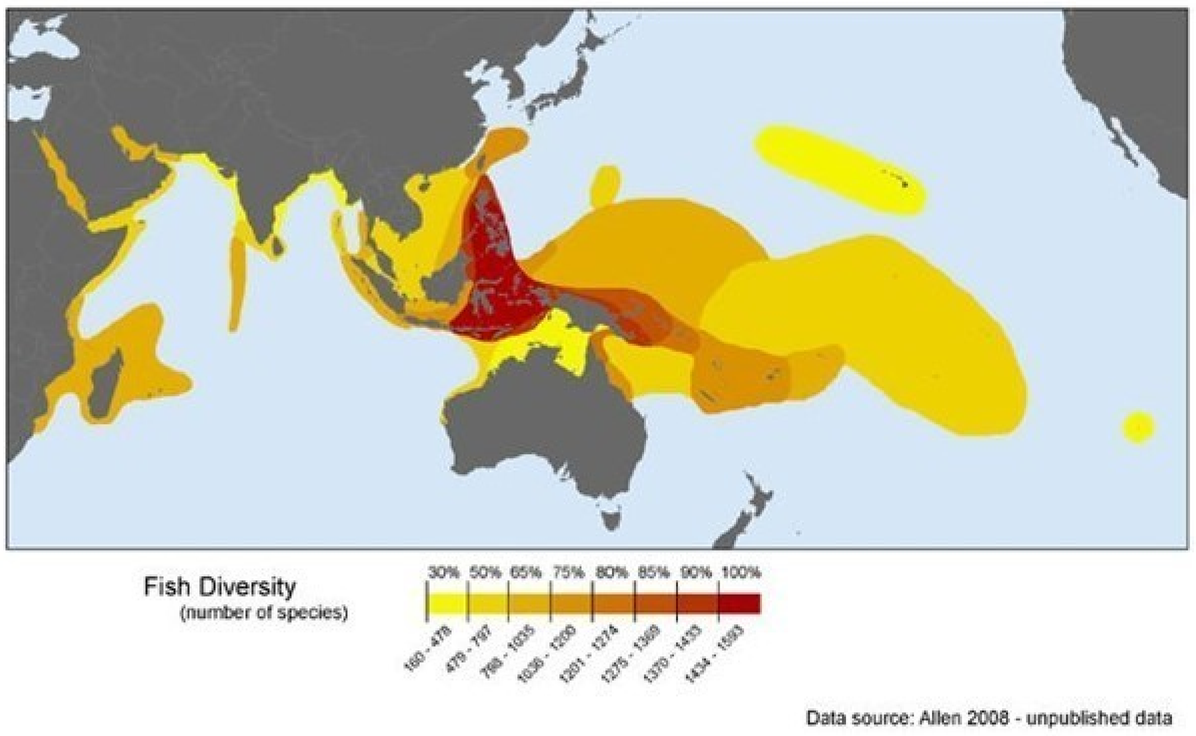


Figure 2: Coral Triangle

```
neralb_CT_meta <- queryMetadata(
  entity = "Sample",
  query = "genus = Nerita AND specificEpithet = albicilla AND decimalLongitude:[ 91 TO 162] AND decimalL
  select = "Event")
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
#Full query is as so:
# query = "genus = Nerita AND specificEpithet = albicilla AND decimalLongitude:[ 91 TO 162] AND
# decimalLatitude:[-11 TO 17] AND _exists_:sequence"

#View(neralb_CT_meta$Sample)

#View(neralb_CT_meta$Event)

    #subset the event metadata
    event_info <- neralb_CT_meta$Event[,c("locality","country","decimalLatitude","decimalLongitude","eve


    kable(event_info)
```

locality	country	decimalLatitude	decimalLongitude	eventID
Lenakel, Tanna	Vanuatu	-5.963366667	105.8646167	6e48e338b3b2ff57c77d8ec5a83dcbcc
Anigua	USA	13.48062	144.73698	065a7d5493f6fe26e1768d9ab37ee9b0
Alam Kotok	Indonesia	-5.69965	106.5381167	e7b1bc2b09b17cdb6348e524fb3f32fd
Yapen-West tip of Serui	Indonesia	-1.905283366	136.2275333	7a9d6b0068b6a550ea4d0e1190096465
Sangiang	Indonesia	-5.963366667	105.8646167	529b51e078402a0dd6b287aaf085a6a0
Cape Gaya	Malaysia	6.1	116.08333	f69dd91bdb0763a2bbb4d55b69c6c1fc
Bak Bak	Malaysia	6.88333	116.83333	6c9d81c99ae5cfb4547a744ece0dddcf
Pantai Kulambu	Malaysia	6.36667	116.36667	416db41063fa7bbe433df3b4b6a0bfb9
Tubod Mar Bohol	Philippines	9.95	124.01667	a53390bc75157460f2c23597e1e4bb22
Balite Mindoro	Philippines	12.91667	121.46667	50724d762731f5841654c9fbb97064b5
Sabang Mindoro	Philippines	13.51667	120.96667	11c3b9453719ac24a27303ebb5faaffc
Doljo Bohol	Philippines	9.58333	123.71667	5f478c31227c9cc6bc52bc1b5ed04282
Alona Beach Bohol	Philippines	9.83333	124.16667	65fece0fa73f579139041886346ab94d
Huon Peninsula	Papua New Guinea	-6.41667	147.5	4be7b661593277796c64372e3b075467
East Coast Pk	Singapore	1.36667	103.8	f0d16e79aabb85fa53baadc4884ef6a0
Laem Phanwa Phuket	Thailand	7.88333	98.4	299f9c7fc870d10f7d1077cd7ceaa838
Ao Nang	Thailand	8.03333	98.8	bb8e2e634741666cffb245a53f9886cd
Hat Chaweng	Thailand	9.5	100	6e223d93358c37c5208906b81fce702d

For simplicity's sake, let's do a little bit of lumping and clean up the locality names.

```
# delete the one sample from Vanuatu
neralb_CT_meta$Sample <- neralb_CT_meta$Sample%>% filter(!materialSampleID == "268.01")

#Lump all Malaysia populations into "Sarawak" - the Borneo part of Malaysia
neralb_CT_meta$Event$locality[which(neralb_CT_meta$Event$country ==
                                      "Malaysia")] <- "Sarawak"

# Change USA to Guam
neralb_CT_meta$Event$country[which(neralb_CT_meta$Event$country ==
                                     "USA")] <- "Guam"
```

```
#lump all Mindoro populations together
neralb_CT_meta$Event$locality[grep("Mindoro",neralb_CT_meta$Event$locality)]<-"Mindoro"

#lump all Bohol populations together
neralb_CT_meta$Event$locality[grep("Bohol",neralb_CT_meta$Event$locality)]<-"Bohol"

#remove commas from locality names
neralb_CT_meta$Event$locality <- str_replace_all(neralb_CT_meta$Event$locality,
                                              ",","")

#remove spaces from locality and country names
neralb_CT_meta$Event$locality<-str_replace_all(neralb_CT_meta$Event$locality,
                                              " " ,"_")
neralb_CT_meta$Event$country<-str_replace_all(neralb_CT_meta$Event$country,
                                             " " ,"_")

#lump the western Thai populations into "Andaman Sea"
neralb_CT_meta$Event$locality[
  which(as.numeric(neralb_CT_meta$Event$decimalLongitude) < 100)
  ] <- "Andaman Sea"

event_info <- neralb_CT_meta$Event[,c("locality","country",
                                     "decimalLatitude","decimalLongitude",
                                     "eventID")]
```

Genetic Data

Ok, so now it's time to download the genetic data. There are genetic data from several different genes on the mitochondrial genome in GEOME. We are going to download the data from the Cytochrome Oxidase subunit 1 gene, or CO1. This is a gene that is involved in the electron transport chain, so parts of it don't change much between different species (since all eukaryotic species respire). However, it changes just enough over evolutionary time that we can often tell the difference between different species.

Because of this, CO1 has become the "universal" barcoding gene for eukaryotes. Because most organisms shed tissue all over the place (you are shedding skin cells as you read this) we can sample DNA floating around in the ocean, or in the soil (environmental DNA or eDNA) and find out what lives there. Cool! But I digress.

```
#lets see what genetic loci are available in GEOME
listLoci()
```

```
## [[1]]
## [1] "18S"
##
## [[2]]
## [1] "S7"
##
## [[3]]
## [1] "TMO"
##
## [[4]]
## [1] "GnRH"
##
## [[5]]
## [1] "NEW_MARKER"
```

```
##
## [[6]]
## [1] "ND2"
##
## [[7]]
## [1] "16S"
##
## [[8]]
## [1] "CR"
##
## [[9]]
## [1] "CO1"
##
## [[10]]
## [1] "CO2"
##
## [[11]]
## [1] "CYB"
##
## [[12]]
## [1] "RAG"
##
## [[13]]
## [1] "A68"
```

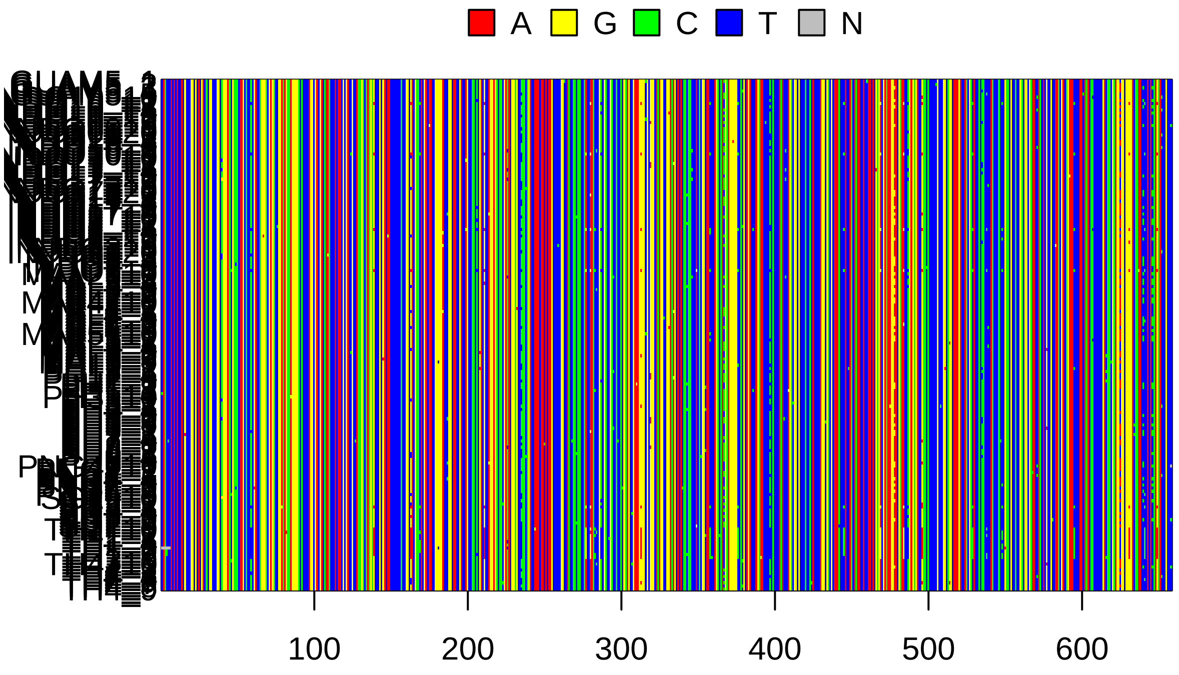
```
#now we just repeat our query, but download FASTA data instead of the metadata
neralb_CT_CO1 <- querySanger(
query = "genus = Nerita AND specificEpithet = albicilla AND decimalLongitude:[ 91 TO 162] AND decimalLa
locus = "CO1")

#Full query is as so:
# query = "genus = Nerita AND specificEpithet = albicilla AND decimalLongitude:[ 91 TO 162] AND
# decimalLatitude:[-11 TO 17] AND _exists_:sequence"

neralb_CT_CO1
```

```
## 163 DNA sequences in binary format stored in a list.
##
## All sequences of same length: 658
##
## Labels:
## https://n2t.net/ark:/21547/BRb2268.01_CO1 [marker = CO1] [ti...
## https://n2t.net/ark:/21547/BRb2GUAM5_1_CO1 [marker = CO1] [t...
## https://n2t.net/ark:/21547/BRb2GUAM5_2_CO1 [marker = CO1] [t...
## https://n2t.net/ark:/21547/BRb2GUAM5_3_CO1 [marker = CO1] [t...
## https://n2t.net/ark:/21547/BRb2GUAM5_4_CO1 [marker = CO1] [t...
## https://n2t.net/ark:/21547/BRb2GUAM5_5_CO1 [marker = CO1] [t...
## ...
##
## Base composition:
##     a     c     g     t
## 0.206 0.168 0.219 0.408
## (Total: 107.25 kb)
```

```r
# Hmmm...we got a few more sequences here than what we have metadata for.
#Let's do some regular expression kung fu to pull out just the sample names
#from the fasta file, and overwrite the names with these shorter values.

names(neralb_CT_CO1) <- str_replace(string=names(neralb_CT_CO1),
                                   pattern=".+\\[tissueID = (.+?)\\].+",
                                      replacement = "\\1")

#missing_metadata<-setdiff(names(neralb_CT_CO1), #neralb_CT_meta$Sample$materialSampleID)

#lets just keep the sequences which we have data for
neralb_CT_CO1 <- neralb_CT_CO1[which(names(neralb_CT_CO1) %in%
                                      neralb_CT_meta$Sample$materialSampleID)]
```

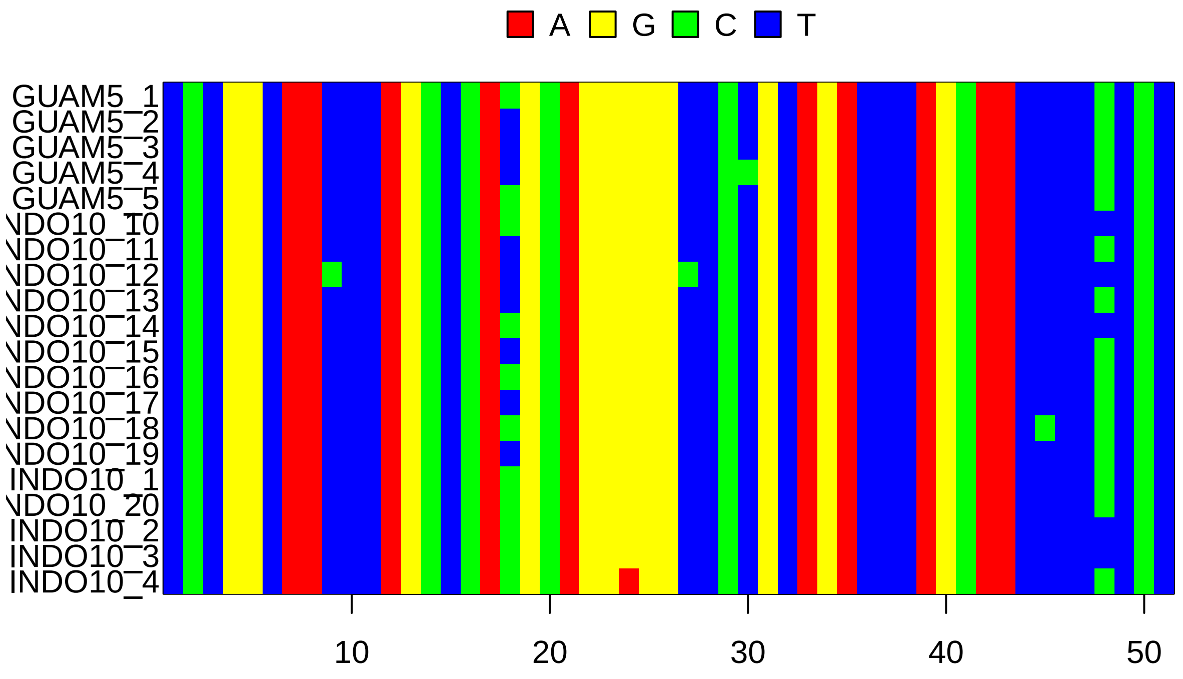
Sequence Alignment Need to align the data to ensure that each site is homologous.

```r
# neralb_CT_CO1 <- as.matrix(neralb_CT_CO1)
neralb_CT_CO1 <- muscle(neralb_CT_CO1)
image.DNAbin(neralb_CT_CO1)
```



Ooh, all kinds of good variation in there!

```r
image.DNAbin(neralb_CT_CO1[1:20,350:400])
```



Merging Metadata with Genetic Data

Storing data in first normal form is all fine and good, but we need to find out what locality each of these individuals was sampled at. So we need to make a non-efficient "flat" table where we **join** the two tables together. We can use the **eventID** key that is common to both tables to do so. Whenever R encounters a particular **eventID** in the Sample table, it will look up the associated information in the Event table and add it to that line of the sample table. We do a left join, so that it will add event information to every line of the Sample table (the Sample table is on the left in the command below).

```
neralb_flat <- left_join(neralb_CT_meta$Sample, neralb_CT_meta$Event, by = "eventID")
```

Now we want to make sure that the sequences are in the same order as the metadata. The `materialSampleID` field in the metadata should be the same as the `rowname` in the sequence file.

```
head(neralb_flat$materialSampleID)
```

```
## [1] "GUAM5_1"   "GUAM5_2"   "GUAM5_3"   "GUAM5_4"   "GUAM5_5"   "INDO10_1"
```

```
head(rownames(neralb_CT_CO1))
```

```
## [1] "GUAM5_1"    "GUAM5_2"    "GUAM5_3"    "GUAM5_4"    "GUAM5_5"    "INDO10_10"
```

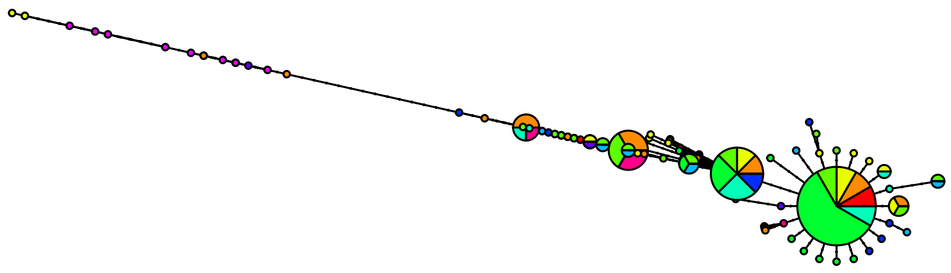
```
neralb_flat <- neralb_flat[order(neralb_flat$materialSampleID),]

neralb_CT_CO1 <- neralb_CT_CO1[order(rownames(neralb_CT_CO1)),]
```

Haplotype Network

Just as we did with the dogs. A haplotype network shows the relationships between haplotypes (with edges or lines), and the frequency of haplotypes (with the size of each circle) in the dataset. Unlike a phylogeny, it does not try to depict ancestral relationships, but instead assumes that the ancestral sequence is one of the other haplotypes.

```
# Let's make a list of populations that were sampled with the locality and country name
pop <- paste(neralb_flat$country,neralb_flat$locality,sep="_")
```

```r
d <- dist.dna(neralb_CT_CO1)
h <- pegas::haplotype(neralb_CT_CO1)
h <- sort(h, what = "label")

#Create the network
net <- pegas::haploNet(h)

#Funky code to make a table of which haplotypes occur in which populations
i <- stack(setNames(attr(h, "index"), rownames(h)))
i <- i[order(i$values),]
ind.hap <- table(hap=i$ind, pop=pop)

# Let's make a second, simpler, table that
#just distinguishes Indian Ocean vs. Pacific Ocean populations
pop2 <- pop

#Change all entries with "Thailand_Andaman Sea" to "Indian Ocean"
pop2[which(pop2=="Thailand_Andaman Sea")] <- "Indian Ocean"
# Change all others to "Pacific Ocean"
pop2[which(pop2!="Indian Ocean")] <- "Pacific Ocean"
#make the table
ind.hap2 <- table(hap=i$ind, pop=pop2)


#play with scale.ratio to get appropriate branch lengths
plot(net,size=attr(net, "freq"), scale.ratio=2, pie=ind.hap,legend=F,
     labels=F,threshold=0, show.mutation=2, fast = T)
```



```r
#legend("topleft", colnames(ind.hap), col=rainbow(ncol(ind.hap)), pch=19, ncol=2)
```

If the resulting figure doesn't look good (and it probably won't) - try this program called PopArt.

Preparing for PopArt:

1. Do the following in R

```r
hapseqs<-h[] # pull out the DNAbin object from the haplotype object
#write this to nexus format
write.nexus.data(hapseqs,
                 file="neralb_haplotypes.nex",
                 interleaved=F)
#write both haplotype tables to csv
write.csv(ind.hap,
          file="neralb_table.csv",
          quote=F)
```

```
write.csv(ind.hap2,
          file="neralb_table_IP.csv",
          quote=F)

#write the flattened metadata too.
write.csv(event_info,
          file="neralb_event_metadata.csv",
          quote=F)
```

2. Open the csv file in Excel. You may want to move the columns into geographical order of some kind.
3. Modify the output csv file in a text editor like Atom, removing the first comma on the first line. Also remove any commas from any population names.

Using PopArt

1. Download and open the program
2. File –> Open –> Select your .nex file. Check that it imported correctly. You should see a matrix of DNA data.
3. File –> Import –> Traits –> Select your modified .csv file. Do not clear the alignment data. On the next screen, verify that the data are being imported correctly (compare to the actual text file)
4. Network –> Median Joining Network. It will compute a median joining network, which combines all possible Minimum Spanning Treesfor the data, inferring missing haplotypes along the way.

a. Minimum Spanning Trees are networks that minimize the total distance of edges (lines) between nodes. If you were trying to connect a bunch of houses with electrical cable on a budget, you might select the minimum spanning network.

5. File –> Save As.. save the nexus file with "_table" appended to the filename.
6. Open this new file using Github Atom or another text editor. You'll notice that PopArt has added a traits block of code that includes the table that you imported. Add the following two space-delimited lines to this block, between the *Format* and the *TraitLabels* lines. When working with your own data, you will need to look up the lat and long data from your event metadata and add the values in the same order as the populations are listed in the TraitLabels line. Save the file.

```
TraitLatitude 13.48062 -5.69965 -5.963366667 -1.905283366 6.1
-6.41667 9.95 13.51667 1.36667 7.88333 9.5 -5.963366667;
TraitLongitude 144.73698 106.5381167 105.8646167 105.8646167 116.08333
147.5 124.01667 120.96667 103.8 98.4 100 105.8646167;
```

7. Re-open the modified file in PopArt. In the network view, in the key, right-click on the "Indian Ocean" population and tell PopArt to color haplotypes from this population black. Right-click on the "Pacific Ocean" and color haplotypes from this population white.

Diversity Statistics

We now need to generate some statistics that tell us how diverse our data are.

Haplotypes and Haplotype Diversity Diversity has two components. The first is richness, or how many different types are there… in this case haplotypes. The second is evenness, or how the different types are distributed. For example an all-white school with 1 Latinx person, one African-American and 1 Asian would be less even (and thus less diverse) than your current institutions, even if the richness was the same.

If we drew two haplotypes at random from the population, haplotype diversity expresses the probability that they would be different haplotypes.

Remember that:

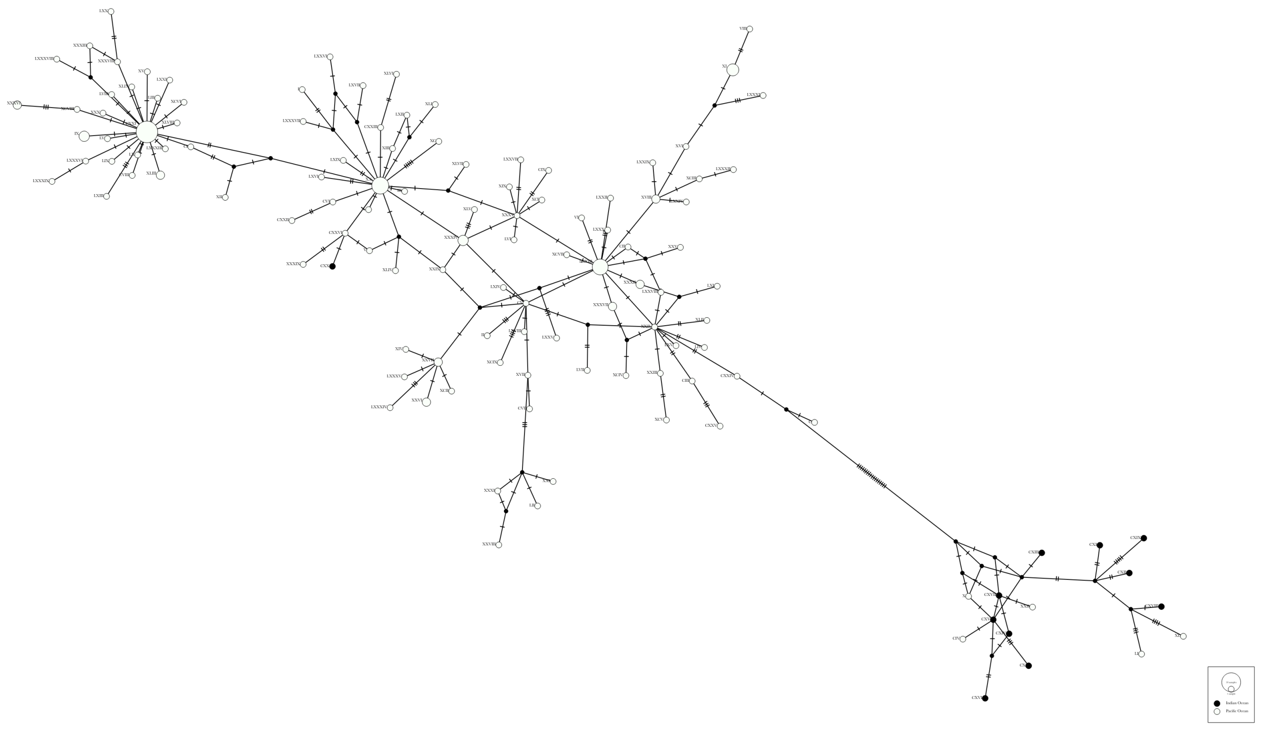


Figure 3: Haplotype Network

$$h = \frac{N}{N-1}(1 - \sum_{i=1}^{n} X_i)$$

Where X_i is the relative haplotype frequency of each haplotype in the sample, and N is sample size.

For this we need the pegas package, and we need to write our own function, because the `hap.div()`and `nuc.div()` functions don't stratify the data by population.

```
# Let's create a copy of our dataset
neralb_CT_C01_b <- neralb_CT_C01

rownames(neralb_CT_C01_b)<-pop

stratastat <- function(x,pop=pop,fun=nuc.div){
  #this function will calculate stats for a DNAbin object
  #(x), stratified across populations given in pop.
  # Some functions this will work with: nuc.div(),
  # theta.s(), tajima.test() from pegas, FusFs(), exptdHet() from strataG,
stats<-NULL
for(p in unique(pop)){
  stat<-fun(x[grep(p,rownames(x)),])
  stats<-c(stats,stat)
}


names(stats) <- unique(pop)
return(stats)
}
```

```
#let's try it out
hapdivs <- stratastat(neralb_CT_CO1_b,pop=pop,fun=hap.div)

hapdivs
```

```
##                         Guam_Anigua              Indonesia_Alam_Kotok
##                           1.0000000                         0.9883041
## Indonesia_Yapen-West_tip_of_Serui                 Indonesia_Sangiang
##                           0.9941520                         1.0000000
##                     Malaysia_Sarawak              Philippines_Mindoro
##                           0.9458128                         1.0000000
##                    Philippines_Bohol   Papua_New_Guinea_Huon_Peninsula
##                           1.0000000                         0.9777778
##              Singapore_East_Coast_Pk              Thailand_Andaman Sea
##                           1.0000000                         1.0000000
##                 Thailand_Hat_Chaweng
##                           0.9722222
```

Nucleotide Diversity Remember that this is the average number of pairwise differences is between any two sequences. Recalling that:

$$\pi = \sum_{ij} x_i x_j \pi_{ij}$$

Where x_i and x_j are the respective frequencies of the ith and jth sequences in the sample and π_{ij} is the number of nucleotide differences between these two sequences.

```
nucdivs<-stratastat(neralb_CT_CO1_b,pop=pop,fun=nuc.div)
nucdivs
```

```
##                         Guam_Anigua              Indonesia_Alam_Kotok
##                         0.009118541                       0.014610996
## Indonesia_Yapen-West_tip_of_Serui                 Indonesia_Sangiang
##                         0.008620843                       0.016369668
##                     Malaysia_Sarawak              Philippines_Mindoro
##                         0.007830865                       0.010371088
##                    Philippines_Bohol   Papua_New_Guinea_Huon_Peninsula
##                         0.010797227                       0.008929477
##              Singapore_East_Coast_Pk              Thailand_Andaman Sea
##                         0.014994934                       0.014947016
##                 Thailand_Hat_Chaweng
##                         0.009202972
```

Here's some optional code to convert our data to be used by the StrataG package. We can use this to double-check the calculations that were done by Pegas.

```
neralb_g <- sequence2gtypes(neralb_CT_CO1,strata=pop)
neralb_g <- labelHaplotypes(neralb_g)
```

```
## Warning: The following samples are missing data for all loci and have been
## removed: PNG4_11, TH1_6
```

```
samples<- numGenotyped(neralb_g, by.strata=T)
# if strataG isn't working use this:
# samples <- neralb_g@data %>% count(stratum)
```

```
haplotypes <- numAlleles(neralb_g, by.strata=T)
```

Now let's make a table of our various summary statistics.

```
diversities<-cbind("# Samples" = samples$num.genotyped,
                   "# Haplotypes" = haplotypes$num.alleles,
                   "Nucleotide Diversity" = nucdivs,
                   "Haplotype Diversity" = hapdivs)

kable(diversities, digits=3)
```

	# Samples	# Haplotypes	Nucleotide Diversity	Haplotype Diversity
Guam__Anigua	5	5	0.009	1.000
Indonesia__Alam__Kotok	19	17	0.015	0.988
Indonesia__Yapen-West__tip__of__Serui	19	18	0.009	0.994
Indonesia__Sangiang	18	18	0.016	1.000
Malaysia__Sarawak	29	22	0.008	0.946
Philippines__Mindoro	18	8	0.010	1.000
Philippines__Bohol	14	18	0.011	1.000
Papua__New__Guinea__Huon__Peninsula	9	14	0.009	0.978
Singapore__East__Coast__Pk	10	10	0.015	1.000
Thailand__Andaman Sea	10	10	0.015	1.000
Thailand__Hat__Chaweng	9	8	0.009	0.972

F-Statistics with DNA data

Thus far we have learned about calculating F_{ST} under the infinite sites model, i.e., we consider each allele or haplotype to be different from each other allele or haplotype, and always different by the same amount (whether that is one mutation or twenty mutations). We have measured genetic diversity using heterozygosity.

What if we could measure the number of differences between each allele (or haplotype)? With DNA data we can! How can we incorporate this additional information into a similar measure of subpopulation differentiation?

Using π, we can define a similar measure of population differentiation as F_{ST}, but this time using a measure of nucleotide diversity (π) within a population, in place of heterozygosity (H) or haplotype diversity (h).

If we define π_T as the average nucleotide diversity in the total sampled populations, and π_S as the average nucleotide diversity within subpopulations, then we can derive a familiar expression for an Fst –like nucleotide measure of subpopulation differentiation:

$$\Phi_{ST} = \frac{\pi_T - \pi_S}{\pi_T}$$

This statistic gives us the proportion of nucleotide diversity among subpopulations, relative to the total – and their values are usually quite similar, particularly with larger sample sizes. The same things can be said for all the different ways of calculating $F_{ST}$from allelic data. Given the multitude of different descriptor variables used by different authors, and the fact that within the two classes they are trying to estimate essentially the same parameters, my convention is to refer to the allelic form of the statistic as F_{ST} , and the nucleotide diversity form as Φ_{ST} , and simply mention somewhere whose formulae or program you used to calculate them.

Pairwise Statistics on Our Data

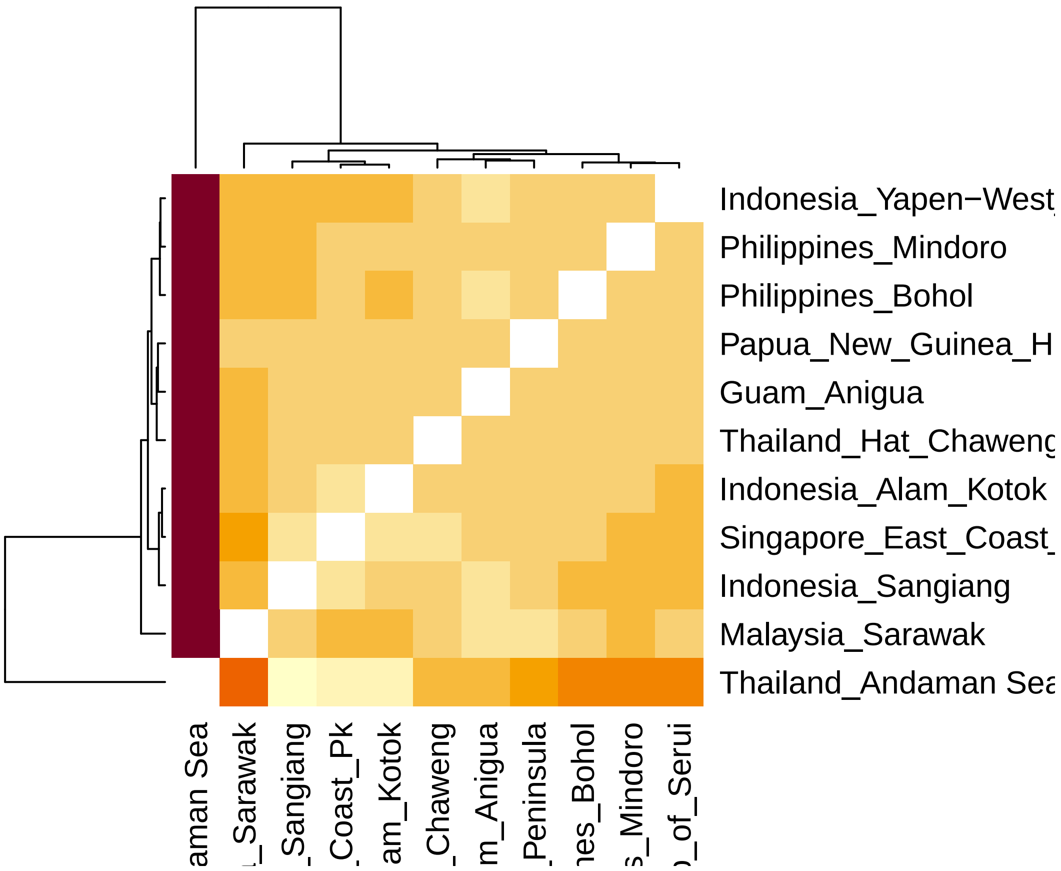
For our purposes now, we will first be calculating **pairwise** Φ_{ST}. For pairwise F-statistics, we are taking every pair of populations, and calculating π_S and π_T and getting Φ_{ST} for each pair. In this way, we create a distance matrix of genetic distance between each pair of populations.

The StrataG package is establishing significance of each pairwise value by randomly allocating each haplotype sequence in the pair of populations being compared into one of two bins, and measuring Φ_{ST} between them. This creates a distribution of Φ_{ST} values that would occur if there was no real structuring going on. If the true Φ_{ST} value is larger than 95% of values in this distribution, then it is deemed significantly greater than zero.

```r
pairwise_phi <- pairwiseTest(neralb_g,nrep=1000,quietly=T,model="raw")

pairwise_phi_mat <- pairwiseMatrix(pairwise_phi, stat = "PHIst")
# PhiST in lower triangle, p-value in upper triangle
kable(pairwise_phi_mat, digits=3)
```

	Guam_Anigua	Indonesia_Alam_Kotok	Indonesia_Sangiang	Indonesia_Yapen-West_tip_of_Serui	Malaysia_Sarawak	Papua_New_Guinea_Huon_Peninsula	Philippines_Bohol	Philippines_Mindoro	Singapore_East_Coast_Pk	Thailand_Andaman Sea	Thailand_Hat_Chaweng
Guam_Anigua	NA	0.401	0.703	0.682	0.345	0.778	0.907	0.512	0.460	0.002	0.710
Indonesia_Alam_Kotok	-0.017	NA	0.559	0.087	0.006	0.309	0.232	0.215	0.750	0.001	0.627
Indonesia_Sangiang	-0.052	-0.024	NA	0.129	0.082	0.462	0.178	0.203	0.612	0.001	0.509
Indonesia_Yapen-West_tip_of_Serui	-0.063	0.058	0.055	NA	0.092	0.361	0.677	0.442	0.090	0.001	0.419
Malaysia_Sarawak	-0.005	0.119	0.067	0.050	NA	0.495	0.060	0.040	0.018	0.001	0.099
Papua_New_Guinea_Huon_Peninsula	-0.076	0.005	-0.022	-0.003	-0.021	NA	0.525	0.518	0.465	0.001	0.539
Philippines_Bohol	-0.092	0.032	0.047	-0.022	0.067	-0.019	NA	0.588	0.240	0.001	0.822
Philippines_Mindoro	-0.034	0.037	0.050	-0.011	0.103	-0.025	-0.018	NA	0.145	0.001	0.416
Singapore_East_Coast_Pk	-0.027	-0.068	-0.055	0.060	0.134	-0.001	0.026	0.049	NA	0.001	0.866
Thailand_Andaman Sea	0.803	0.719	0.669	0.866	0.891	0.824	0.846	0.842	0.699	NA	0.001
Thailand_Hat_Chaweng	-0.064	-0.036	-0.027	0.001	0.076	-0.032	-0.045	-0.006	-0.057	0.811	NA

```r
phiST <- pairwise_phi_mat
phiST[upper.tri(phiST)] = t(phiST)[upper.tri(phiST)]

heatmap(phiST)
```



and we can see that Φ_{ST} values between the Andaman Sea and elsewhere are high and significant!