Pairwise SNP difference summary

Anders Goncalves da Silva

26 August 2015

The goal

There might come a time when it would be interesting to measure just how different clades are, or how distinct are different classification schemes (e.g., MLST categories). We can use the output from `nullabor`, in particular, the alignment of the SNPs across isolates, to quantify the mean distance (and other quantiles of interest), in terms of distinct SNPs across groups of isolates.

How to do it

We can calculate summary measures of differentiation using `R`. Here, I am providing a script that can be run within `R` or from a command line interface.

Obtaining the script

Please clone the `git-hub` repository.

Running within `R`

To run within `R`, make sure the `ape` library is installed:

```r
install.packages("ape")
```

Once that is done, open the script `pairwise_snp_differences.R` in a text editor. If within `RStudio`, just double click on the script within the `File` pane (usually on the lower right). Then, edit the following lines:

```
################################################################################
# If not running off the command line, change these parameters to point to the
# approriate files, e.g.:
#   cat = '/home/user/cat.csv'
#
#   To test the script substitute below as follows:
#
#     cat_file = 'test/woodm_grouping.csv'
#     seq_file = 'test/woodm.fa'

cat_file = NULL

# Only one of these files needs to be specified. If both are specified, the
# diff file will have precedence
seq_file = NULL
diff_file = NULL
```

```
# Options:
#   Change the following options to set output
#
out_basename = 'snp_diff'
tab_fmt = "csv" # options are "csv" or "md"
tab_type = "pretty" # options "pretty" or "raw" --- "pretty" formats numbers in
                      # scientific format (e.g., 1.05e-9), while "raw" gives
                      # raw value outputs
fig_fmt = "png" # options are "png" or "pdf"
exclude_ids = NULL # a string to a path to a file with one sequence ID per
                   # line. these sequences will be excluded from the
                   # analysis.

################################################################################
```

Then, select the whole script, and click `run`.

From the command-line

If running the script from the command-line, just type:

```
Rscript pairwise_snp_differences.R --help
```

Or, if in Windows:

```
Rscript.exe pairwise_snp_differences.R --help
```

That will give you a run down of the different parameters.

Building the script

Below, I outline how I got to the script. This will allow interested people to ammend, and expand depending on their specific needs.

Load necessary libraries

```r
require(ape) # a basic phylogenetic package used to load sequence data and calculate distances
require(spider) # a DNA barcoding package with some functionality of interest
require(geiger) # a package for macroevolutionary simulation and estimating parameters related to diver
```

If you don't have them, just use the `install.packages()` command to install them:

```r
install.packages("ape")
```

To run the script, you only need the ape package. The others are only necessary if you wish to run this tutorial

Load the sequence data

Here, I am using some example data provided with the package `ape`. The data consists of 15 mitochondrial cytochrome `b` sequences from the woodmouse.

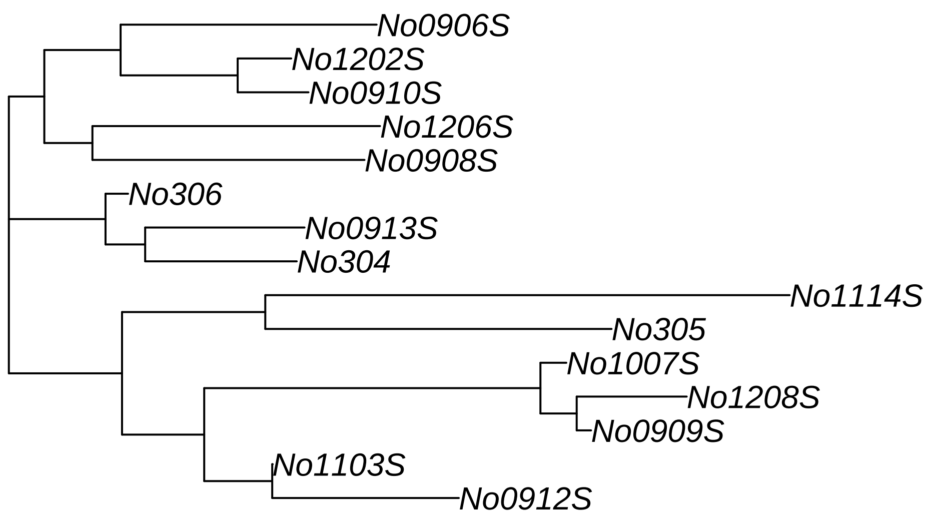
```r
woodm <- read.FASTA(file = "test/woodm.fa") # woodm.fa is a FASTA file containing multiple aligned sequ
```

It is easy to perform some basic tree-building within `R`. For instance:

```r
#calculate distance
woodm_raw_dist <- dist.dna(x = woodm, model = "raw")

#use distance to build a Neighbour-Joining tree
woodm_nj_tree <- nj(X = woodm_raw_dist)

#plot the tree
plot(woodm_nj_tree)
```

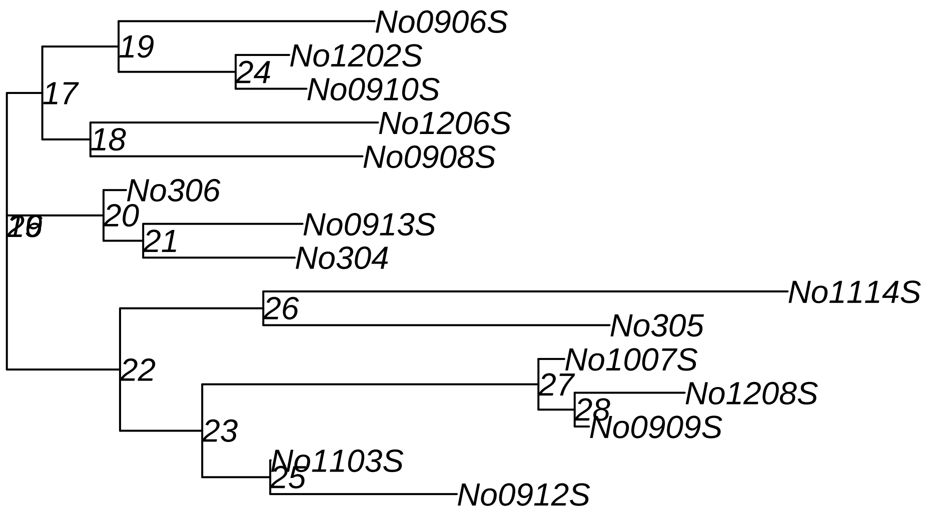


Load metadata

To calculate the statistics of interest, we need some metadata that groups the sequences/isolates into categories. These might be MLST categories, or any other grouping of interest (e.g., isolates could be grouped by geography or year). The grouping categories should be in a `CSV` or `tab` delimited file, with the one column containing the sequence IDs, as in the FASTA file above, and one or more columns corresponding to each of the classification schemes of interest.

For this example, we will calculate summary statistics for the pairwise differences among the three clades identified in the NJ tree above (defined by the three basal branches). I'll first show you how this metadata could be generated from a tree within `R`. But, most likely you will have this information ready from other sources.

```r
# first, we will figure out the basal node for each of the three clades
woodm_nj_tree$node.label<-((length(woodm_nj_tree$tip)+1):((length(woodm_nj_tree$tip)*2)-1))
plot(woodm_nj_tree, show.node.label = T) # as we can see, these are 17, 20, and 22
```



```
# we can then pull out the tips associated with each of basal nodes
clade_a <- tips(phy = woodm_nj_tree, node = 17)
names(clade_a) <- rep("clade_a", length(clade_a))
clade_b <- tips(phy = woodm_nj_tree, node = 20)
names(clade_b) <- rep("clade_b", length(clade_b))
clade_c <- tips(phy = woodm_nj_tree, node = 22)
names(clade_c) <- rep("clade_c", length(clade_c))

#metadata object
metadf <- do.call('rbind', lapply(list(clade_a, clade_b, clade_c), function(group) data.frame(seq_id =

print(metadf)
```

```
##     seq_id   clade
## 1  No0908S clade_a
## 2  No1206S clade_a
## 3  No0910S clade_a
## 4  No1202S clade_a
## 5  No0906S clade_a
## 6    No304 clade_b
## 7  No0913S clade_b
## 8    No306 clade_b
## 9  No0912S clade_c
## 10 No1103S clade_c
## 11 No0909S clade_c
## 12 No1208S clade_c
## 13 No1007S clade_c
## 14   No305 clade_c
## 15 No1114S clade_c
```

In case that your data is already saved in a `CSV` or `tab` delimited file, you can load it with the following:

```
# I have saved the data.frame to a CSV file as an examle: woodm_grouping.csv

# To load a CSV file do the following
```

```r
metadf <- read.table(file = "test/woodm_grouping.csv", sep = ",", header = T)
# if tab-delimited file change sep = ',' to sep = '\t'
```

Now, we can write a function that will take the distance object we calculated above, and return some summary statistics of interest.

```r
summ_distances <- function(categories, dist_obj){
  #dist_obj is a distance object produced by using the dist.dna() function of ape
  #categories is a data.frame with two columns:
  #   - seq_id: that matches the sequence ids in dist_obj
  #   - groups: that assigns the individual seq_ids to a group

  # some sanity checks
  if(class(dist_obj) != 'dist' & class(dist_obj) != 'matrix') {
    stop("dist_obj is not an object of type dist or matrix!
         Please use dist.dna() to create a distance object first OR
         input a CSV file with count of differences produced by
         nullabor")
  }

  if(!is.data.frame(categories)){
    stop("categories must be a data.frame!
         Please create a data.frame with the metadata first.")
  }

  if(ncol(categories) > 2) {
    warning("Number of colums in categories is >2,
            taking the first two columns only")
    categories <- categories[,c(1,2)]
  }

  if(!all(sort(names(categories)) == sort(c("seq_id", "groups")))) {
    warning("The columns of categories do not have names this function
            recognizes. It will assume that the first column contains seq_ids,
            and the second column the relevant categories.")
    names(categories) <- c("seq_id", "groups")
  }

  # calculations
  dat <- as.matrix(dist_obj)
  taxa <- unique(as.character(categories[,'groups']))
  n_taxa <- length(taxa)
  total_comp <- (n_taxa^2 + n_taxa)/2
  out <- data.frame(grp1 = character(total_comp),
                    grp2 = character(total_comp),
                    comp = character(total_comp),
                    N = numeric(total_comp),
                    type = rep("inter-group", total_comp),
                    mu = numeric(total_comp),
                    sd = numeric(total_comp),
                    min_dist = numeric(total_comp),
                    max_dist = numeric(total_comp), stringsAsFactors = F)
  n_comp = 1
  for(i in 1:n_taxa){
    g1 <- taxa[i]
    seq_1 <- as.character(categories[categories$groups == g1, 'seq_id'])
    for(j in i:n_taxa){
      g2 <- taxa[j]
      seq_2 <- as.character(categories[categories$groups == g2, 'seq_id'])
      tmp_dat <- dat[seq_1, seq_2]
      out[n_comp, "grp1"] <- g1
      out[n_comp, "grp2"] <- g2
      out[n_comp, "N"] <- length(tmp_dat)
      out[n_comp, "comp"] <- paste(g1, g2, sep='_')
      if(i == j) {
        if(length(tmp_dat) > 1){
          #if length is one, this results in a empty set.
          #so, added this condition to fix the problem
          tmp_dat <- tmp_dat[lower.tri(tmp_dat)]
        }
        out[n_comp, 'type'] <- 'intra-group'
        out[n_comp, "comp"] <- g1
      }
      if (length(tmp_dat) > 1 & max(tmp_dat) > min(tmp_dat)) {
        out[n_comp, "mu"] <- mean(tmp_dat)
        out[n_comp, "sd"] <- sd(tmp_dat)
        out[n_comp, "min_dist"] <- min(tmp_dat)
        out[n_comp, "max_dist"] <- max(tmp_dat)
      } else {
          out[n_comp, "mu"] <- mean(tmp_dat)
          out[n_comp, "sd"] <- 0
          out[n_comp, "min_dist"] <- min(tmp_dat)
          out[n_comp, "max_dist"] <- max(tmp_dat)
        }
      n_comp = n_comp + 1
      }
  }
  out$type <- factor(out$type, levels = c("intra-group", "inter-group"))
  out$comp <- factor(out$comp, levels = out$comp[order(out$type, out$comp)])
  return(out)
}
```

```
names(metadf) <- c("seq_id", "groups")

results <- summ_distances(dist_obj = woodm_raw_dist, categories = metadf)
```

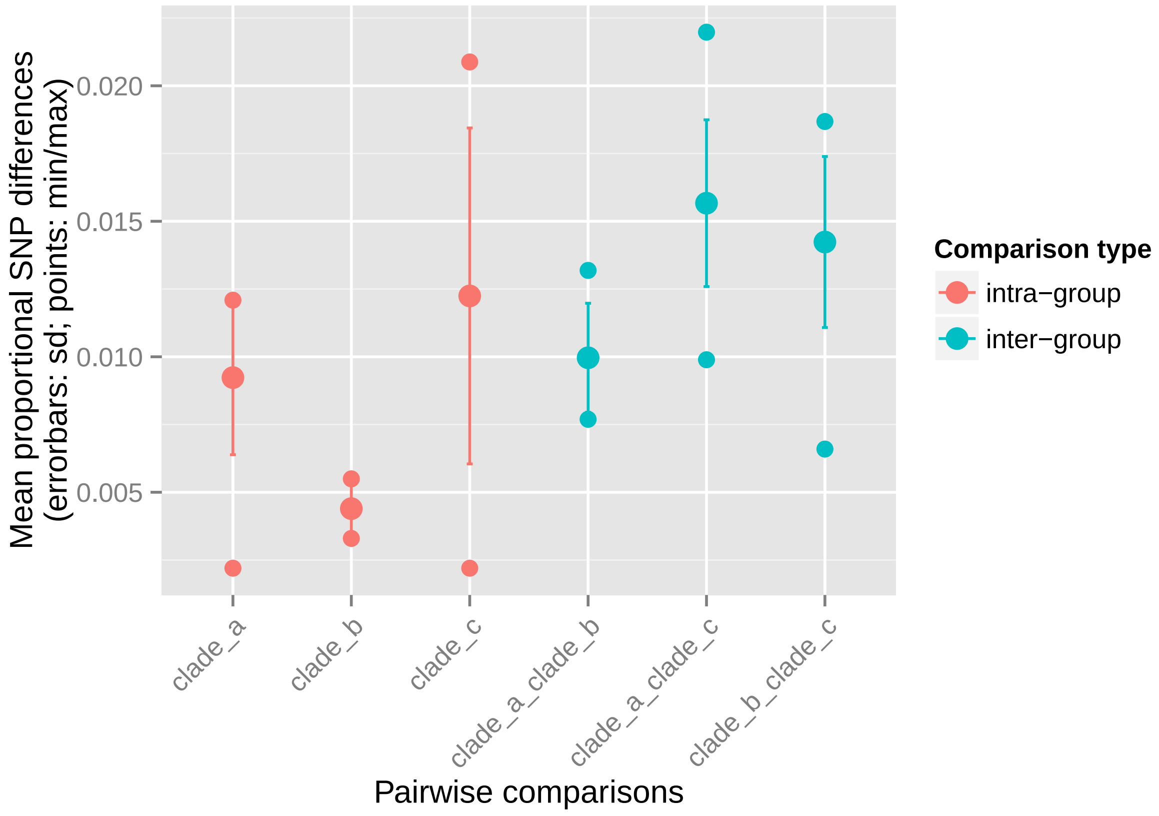
Table 1: Table of summary pairwise SNP differences among groups of woodmouse cytb sequences.

grp1	grp2	N	type	mu	sd	min_dist	max_dist
clade_a	clade_a	25	intra-group	0.009231	0.002847	0.002198	0.01209
clade_a	clade_b	15	inter-group	0.009963	0.002012	0.007692	0.01319
clade_a	clade_c	35	inter-group	0.01567	0.003078	0.00989	0.02198
clade_b	clade_b	9	intra-group	0.004396	0.001099	0.003297	0.005495
clade_b	clade_c	21	inter-group	0.01423	0.003156	0.006593	0.01868

grp1	grp2	N	type	mu	sd	min_dist	max_dist
clade_c	clade_c	49	intra-group	0.01224	0.0062	0.002198	0.02088

We can then plot the results.

```r
require(ggplot2)
ggplot(results, aes(x = comp, y = mu, colour = type)) +
  geom_point(size = 4) +
  geom_errorbar(aes(ymax = mu + sd, ymin = mu - sd, width = 0.05)) +
  geom_point(aes(x = comp, min_dist), size = 3) +
  geom_point(aes(x = comp, max_dist), size = 3) +
  xlab("Pairwise comparisons") +
  ylab("Mean proportional SNP differences\n(errorbars: sd; points: min/max)") +
  scale_colour_discrete(name = "Comparison type") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Additional links of potential interest

A more in-depth tutorial on using R to do phylogenetic-type analyses can be found here.

A more in-depth tutorial on loading and manipulating DNA sequences in R can be found here.

Session info

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.5 (Yosemite)
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_1.0.1  pander_0.5.2   geiger_2.0.3   spider_1.3-0
## [5] pegas_0.8-1    adegenet_2.0.0 ade4_1.7-2     ape_3.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.0      spdep_0.5-88     formatR_1.2      plyr_1.8.3
##  [5] LearnBayes_2.15  tools_3.2.2      boot_1.3-17      digest_0.6.8
##  [9] evaluate_0.7.2   gtable_0.1.2     nlme_3.1-122     lattice_0.20-33
## [13] Matrix_1.2-2     igraph_1.0.1     shiny_0.12.2     DBI_0.3.1
## [17] yaml_2.1.13      parallel_3.2.2   mvtnorm_1.0-3    proto_0.3-10
## [21] coda_0.17-1      dplyr_0.4.2      stringr_1.0.0    knitr_1.11
## [25] grid_3.2.2       deSolve_1.12     R6_2.1.1         rmarkdown_0.8
## [29] sp_1.1-1         reshape2_1.4.1   seqinr_3.1-3     deldir_0.1-9
## [33] magrittr_1.5     scales_0.3.0     htmltools_0.2.6  MASS_7.3-44
## [37] splines_3.2.2    assertthat_0.1   mime_0.3         colorspace_1.2-6
## [41] xtable_1.7-4     httpuv_1.3.3     labeling_0.3     subplex_1.1-6
## [45] stringi_0.5-5    munsell_0.4.2
```

Contact info

Anders Goncalves da Silva (andersgs at gmail dot com).