chromploid Package

Rosana Zenil-Ferguson

2017-01-03

Overview of Chromploid

Chromploid is a package that calculates the negative log-likelihood function for evolution rates of chromosome number or ploidy change models in phylogenetic trees. Currently, chromploid includes three different models of chromosome number or ploidy evolution: BiChroM, PloidEvol, and ChromEvol M3(beta version). Users can choose among these three models but if interested Chromploid package can be extended to accomodate custom chromosome number or ploidy change models.

About the models included

1. BiChroM: A continuous time Markov chain that links the evolution of chromosome number change to a binary trait. This model contains 10 parameters that represent rates of evolution (change per time).[1]
2. BiChroM linear: BETA VERSION. A continuous time Markov chain that links the evolution of chromosome number change to a binary trait where the rates of binary trait change are linear functions of chromosome number. This model is define via 12 parameters that represent rates of evolution (change per time).
3. PloidEvol: A continuous time Markov chain that models the evolution of ploidy change. This model contains 6 parameters that represent rates of ploidy evolution (change per time).
4. ChromEvol M3: A continuous time Markov chain that models the evolution of chromosome number change. The model has 4 parameters that represent rates of chromosome number change. This model was proposed by Mayrose et al. (2010)[2]

Installation

You'll need package `devtools`. Chromploid is easily installed from github

```
library("devtools")
devtools::install_github("roszenil/chromploid")
library("chromploid")
library("geiger") #chromploid depends on geiger so don't forget to load it
```

Chromploid package depends on libraries `ape`, `geiger`, and `expm` so make sure you are running those too.

Negative log likelihood

The function `chromploid_nllike()` is possibly the most important and critical of the package. It calculates the negative log-likelihood of any discrete trait model for chromosome number or ploidy change. Minimizing negative log-likelihoods allows us to infer model parameters, that is, obtaining maximum likelihood estimates (MLEs), profile likelihoods, likelihood-confidence intervals, and likelihood ratio tests necessary to answer biological questions. I will describe how to use this important function after introducing the models coded in chromploid.

[1] Zenil-Ferguson, R., Ponciano, J.M., and Burleigh, J.G. 2016. Testing the association of phenotypes with polyploidy: an example using herbaceous and woody eudicots. Submitted

[2] Mayrose, I., Barker, M.S. and Otto, S.P., 2010. Probabilistic models of chromosome number evolution and the inference of polyploidy. Systematic Biology, 59(2), pp.132-144.

BiChroM

BiChroM -Binary trait associated with chromosome number change model- is a continuous time Markov chain (CTMC) that allows for estimation and testing of chromosomal change rates correlated with a binary trait. The model is defined via 10 parameters that represent rates of chromosomal change linked to a value of a binary trait. BiChroM allows for testing the association of chromosomal change (polyploidy) to phenotypic change.

Description of parameters in BiChroM:

- λ_0, λ_1: Rate of gain in one chromosome for taxa with binary trait 0, or binary trait 1 respectively.
- μ_0, μ_1: Rate of loss in one chromosome for taxa with binary trait 0, or binary trait 1 respectively.
- ρ_0, ρ_1: Rate of chromosome doubling for taxa with binary trait 0, or binary trait 1 respectively.
- q_{01}, q_{10}: Rate of transition between binary state values, from 0 to 1 or vice versa.
- ϵ_0, ϵ_1: Ancillary parameters of chromosomal changes. Rate of chromosome number changes after `size` has been determined.

Full inference for BiChroM model

Inferences using BiChoM model require different functions that allow the calculation of full and partial likelihoods for the parameters. For example, when the goal is to estimate rates, the full likelihood of 10 parameters should be calculated and optimized, but if the goal of the inference is to assess the uncertainty about a single parameter then a univariate likelihood should be the calculated.

Each of the functions listed below creates a Q-matrix, a matrix that defines the rates of chromosomal change linked to the binary trait. The difference among these functions is that each one builds the necessary mathematics to create a different likelihood function, thus, allowing for different types of inference.

- `Q_bichrom()`: Creates the Q-matrix (instantaneous probability rate matrix) for the full model, that allows for calculation of maximum likelihood estimates.
- `Q_unichrom()`: Creates the marginal Q-matrix necessary for calculation of univariate profile likelihoods. Used to assess the uncertainty surrounding an individual parameter.
- `Q_bibichrom()`: Creates the marginal Q-matrix necessary for calculation of bivariate profile likelihoods.Used to assess the uncertainty of two parameters at a time, allows to detect problems with parameter identifiability.[3]
- `Q_reducedbichrom()`: Creates the marginal Q-matrix necessary for calculation of likelihood ratio tests.
- `bichrom_dataset()`: Transforms a data set of chromosome and binary traits to input in the functions above

2. BiChroM simulation and estimation

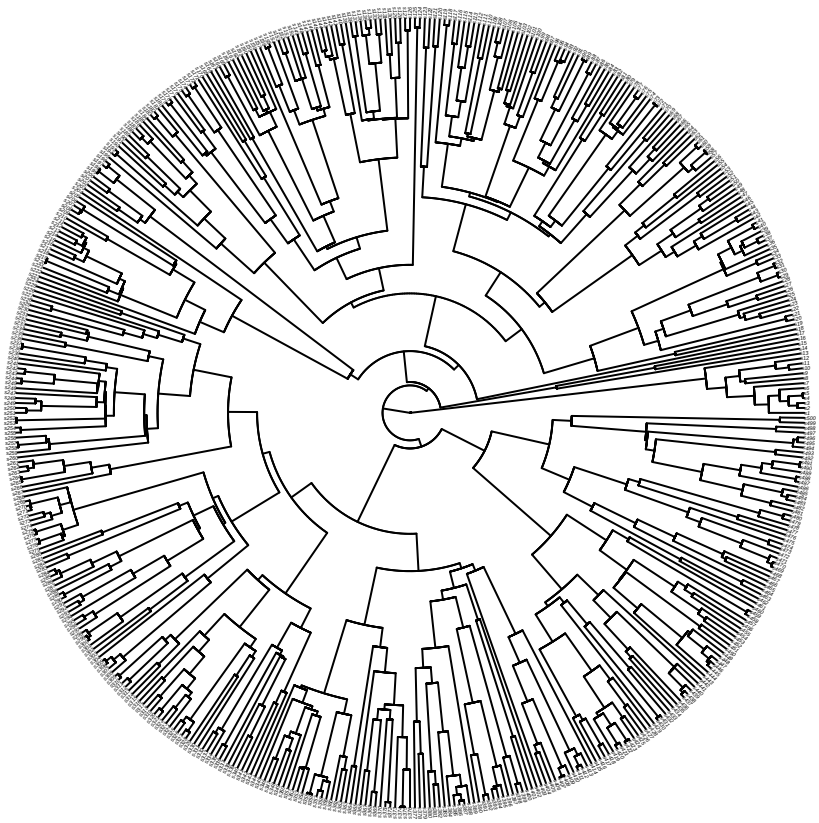
- Choose a set of fixed parameter values. For example $(\lambda_0, \lambda_1, \mu_0, \mu_1, \rho_0, \rho_1, q_{01}, q_{10}, \epsilon_0, \epsilon_1) = (0.12, 0.001, 0.25, 0.002, 0.036, 0.006, 0.04, 0.02, 1.792317852, 1.57e-14)$
- Calculate BiChroM Q-matrix All parameters in BiChroM are used in log scale because numerical calculations are easier than in original scale.

```
log.params <- log(c(0.12, 0.001, 0.25, 0.002, 0.036, 0.006, 0.04,
    0.02, 1.792317852, 1.57e-14))
N <- 50
mymatrix <- Q_bichrom(log.theta = log.params, size = N)
```

[3]Ponciano, J.M., Burleigh, J.G., Braun, E.L. and Taper, M.L., 2012. Assessing parameter identifiability in phylogenetic models using data cloning. Systematic biology, p.sys055.

- Simulate a tree with 500 taxa and BiChroM values Assuming that the root of the tree has a value 56 means that the root is has 5 haploid chromosomes and its binary type is 1.

```
mytree<- sim.bdtree(b=0.055, stop="taxa", n=500, seed=1)
# Seed was fixed to obtain always the same tree for this tutorial
plot(mytree,type="fan",cex=0.2, no.margin=TRUE)
```



```
set.seed(1) #Seed was fixed to obtain always the same sample for this tutorial
mysample<- chromploid_sim.char(phy=mytree, par=mymatrix,model="discrete", root=56)
```

- Calculate the likelihood of the tree, assuming that the root has a uniform distribution for chromosome number and binary state `pi.0=NULL` (other options for the root are available, like Maddison and Fitzjohn (2009)). The value of the negative log-likelihood `nllike` in this example is 943.9293, since we fixed the random seed, and it is calculated as follows.

```
nllike <- chromploid_nllike(log.par = log.params, phy = mytree,
    tip.values = mysample, pi.0 = NULL, Q.FUN = Q_bichrom, Q.ARGS = list(size = 50))
nllike
```

```
## [1] 943.9293
```

If you have the following error appearing

```
# Error in tip.values[i, charnum + 1] : incorrect number of
# dimensions
```

this indicates that you don't have in the first column the species names and in the second column the sample for BiChrom. An easy way to fix it is converting your sample into a data frame with first column having taxa names and second column having the sample needed in BiChroM just as `chromploid_sim.char` does.

```
mysample <- data.frame(taxa = rownames(mysample), sample = as.numeric(mysample))
head(mysample)
```

3. BiChrom optimization Usually you don't know the values of the parameters and what you need is toto find the maximum likelihood estimates of the parameters determined by BiChroM. In this case, you can optimize (minimize) the negative log-likelihood with any R optimizer given that you have a phylogenetic tree and a sample of chromosome numbers and binary trait. I suggest using package `nloptr` because it makes very accurate searches in the parametric space. The cost of using `nloptr` is computational time. You can start with a rough search using `optim` function from `stats` to localize possible values of the parameters close to the global optimum. Here's an example using `nloptr`.

```
# Optimizations take hours to run, I suggest you run this
# routine in a cluster. For a 4700 tip tree starting with a
# x.0 close to the maximum it can take up to 48 hrs.

library(nloptr)
x.0 <- log(c(0.12, 0.001, 0.25, 0.002, 0.036, 0.006, 0.04, 0.02,
    1.792317852, 1.57e-14))
# value where the optimization algorithm is going to start. A
# vector of 10 values for the parameters in log scale
results <- rep(0, 11)  #vector useful to save optimization results
my.options <- list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-08,
    print_level = 1, maxtime = 1.7e+08, maxeval = 1000)
# Options used in nloptr, for more information ?nloptr. This
# is a subplex algorithm, with high tolerance to fine a very
# precise optimum.
mle <- nloptr(x0 = x.0, eval_f = chromploid_nllike, opts = my.options,
    phy = mytree, tip.values = mysample, pi.0 = NULL, Q.FUN = Q_bichrom,
    Q.ARGS = list(size = 50))
print(mle)
results[1:10] <- exp(mle$solution)  # Maximum likelihood estimates in original scale, since we entered
results[11] <- mle$objective  # Negative log-likelihood value at the MLEs (minimum)
print(results)
```

The vector `results` contains in the first 10 entries maximum likelihood estimates of parameters $(\lambda_0, \lambda_1, \mu_0, \mu_1, \rho_0, \rho_1, q_{01}, q_{10}, \epsilon_0, \epsilon_1)$. Do not expect good estimates for ϵ_0 and ϵ_1 since by definition these are ancillary parameters that prevent biases for the other 8 that are key to understand chromosomal change (see Zenil-Ferguson et al. for details). The last entry in `results` is the value of the negative log-likelihood at the maximum likelihood estimates and it is a useful when you want to plot relative profile likelihoods or calculate likelihood ratio tests.

4. Likelihood Ratio Test To calculate the likelihood of the reduced model needed for likelihood ratio tests you will need to use the function `chromploid_nllike` with Q-matrix `Q_reducedbichrom()`. The example here tests if the rates associated with chromosome doubling under the different values of the binary trait are equal that is $H_0 : \rho = \rho_0 = \rho_1$

```
x.0 <- log(c(0.12, 0.001, 0.25, 0.002, 0.01, 0.04, 0.02, 1.792317852,
    1.57e-14))
# Value where the optimization algorithm is going to start. A
# vector of 9 values for the parameters in log scale. The
# value of the hypothesis here is rho=0.010
results.reduced <- rep(0, 10)  # Vector useful to save optimization results
my.options <- list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-08,
    print_level = 1, maxtime = 1.7e+08, maxeval = 1000)
# Options used in nloptr, for more information ?nloptr. This
# is a subplex algorithm, with high tolerance to fine a very
# precise optimum.
mle.reduced <- nloptr(x0 = x.0, eval_f = chromploid_nllike, opts = my.options,
    phy = mytree, tip.values = mysample, pi.0 = NULL, Q.FUN = Q_reducedbichrom,
    Q.ARGS = list(size = 50, equal.param = c("rho0", "rho1"),
        location.par = 5))
# Q.ARGS is a list that has all the arguments included in
# Q_reducedbichrom Type ?Q_reducedbichrom to see a longer
# explanation of each argument size= maximum number of
# haploid chromosome numbers to consider in your sample,
# equal.params=which parameters are equal based on the
# hypothesis H0, location.par is the position in the vector
# where rho value appears
print(mle.reduced)
results.reduced[1:9] <- exp(mle.reduced$solution)
# Maximum likelihood estimates in original scale
results.reduced[10] <- mle.reduced$objective
# Negative log-likelihood value at the MLEs (minimum)
print(results.reduced)
```

Finally the value of the likelihood ratio test for $H_0 : \rho_0 = \rho_1$ hypothesis is simply $LRT = 2 \times$ (full model nloglike – reduced model nloglike) that has a $\chi^2_{(1)}$ distribution

```
LRT <- 2 * (results[11] - results.reduced[10])
p.value <- pchisq(LRT, df = 1)
```

5. Profile likelihoods

Point estimations are useful but so are interval estimations. Profile likelihoods allow for better understanding of the precision, estimability, and identifiability of estimates, given BiChroM model, the sample size, and the phylogenetic tree.

- Univariate profile likelihoods To understand the behavior of one parameter of a time under BiChroM model the function you need is `Q_unibichrom`. Assume that you want to see the profile likelihood of parameter ρ_0. You have to choose a grid (range of values) for which the likelihood could be interesting. I suggest again that you do this procedure in a cluster. In the example here shown, I do a sequential way of calculating the profile in this example but more efficiently you could program this in parallel, sending one optimization per processor, retrieving all the results in 24hrs, so you don't have to wait for months. I also suggest using as your starting point x.0 the maximum likelihood value (in log-scale) that you obtained from optimizing `Q_bichrom`, optimizations will take shorter if your starting point is good.

```
logrho0.values <- log(seq(0.02, 0.04, 0.005))
# Calculate the likelihood values in a grid 0.02, 0.025, ...,
# 0.035, 0.04 (in log scale). True value is around 0.36 so
# values surrounding the true value are interesting
long <- length(logrho0.values)
x.0 <- log(c(0.12, 0.001, 0.25, 0.002, 0.006, 0.04, 0.02, 1.792317852,
    1.57e-14))
# value where the optimization algorithm is going to start. A
# vector of 9 values for the nuisance parameters in log scale
# (everything except rho0)
profile.values <- rep(0, long)  # Vector useful to save optimization results
my.options <- list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-08,
    print_level = 1, maxtime = 1.7e+08, maxeval = 1000)
# Options used in nloptr, for more information ?nloptr. This
# is a subplex algorithm, with high tolerance to fine a very
# precise optimum.
for (i in 1:long) {
    mle.profilerho0 <- nloptr(x0 = x.0, eval_f = chromploid_nllike,
        opts = my.options, phy = mytree, tip.values = mysample,
        pi.0 = NULL, Q.FUN = Q_unibichrom, Q.ARGS = list(log.theta0 = logrho0.values[i],
            size = 50, param.name = "rho0"))
    # Q.ARGS is a list that has all the arguments included in
    # Q_unibichrom Type ?Q_unibichrom to see a longer explanation
    # of each argument
    print(mle.profilerho0)
    profile.values[i] <- mle.profilerho0$objective
}

plot(exp(logrho0.values), profile.values, xlab = expression(rho_0),
    ylab = "Negative log-likelihood", type = "l", lwd = 2)
```

- Bivariate profile likelihood

Analogous to the example above but using a grid for two parameters and calculating using the function `Q_bibichrom`.

BiChroM Linear

THIS IS A BETA VERSION BiChroM linear assumes the same chromosome number changes than BiChroM but the binary trait change is a linear function of chromosome number. This model can be useful when trying to associate if polyploidy is associated with the speed of binary trait change. The model is defined via 12 parameters, chromosomal change rates $\lambda_0, \lambda_1, \mu_0, \mu_1, \rho_0, \rho_1$ are exactly the same as in the simple BiChroM but binary trait change rates are defined as follows.

- $\kappa_0 + iq_{01}$: Chromosome number i dependent transition rate from state 0 to state 1.
- $\kappa_1 + iq_{10}$: Chromosome number i dependent transition rate from state 1 to state 0.

Usage

Substitute `Q_bichrom()` for `Q_bichromlinear()` but the arguments for this new function are a vector log.par of size 12, with parameters $\lambda_0, \lambda_1, \mu_0, \mu_1, \rho_0, \rho_1, \kappa_0, \kappa_1, q_{01}, q_{10}, \epsilon_0, \epsilon_1$ in this order and in log-scale, and `size` the maximum number of haploid chromosome numbers allowed.