Example Solanum

Rosana Zenil-Ferguson

2/16/2018

Libraries needed for the Solanum example

```
library("geiger")
library("nloptr")
library("chromploid")
```

Definition of full and reduced models via Q-matrix

The full model of Solanum is defined via a Q-matrix with six parameters. Two polyploidy rates (ρ_H, ρ_W), two demiploidy rates (ϵ_H, ϵ_W), and two transition rates between binary traits (q_{HW}, q_{WH}). The model has 5 states for herbaceous and 5 states for woody because chromosome numbers in the sample aree 12, 18, 24, 36, and 48. The Q-matrix is simply defined as follows:

```
Q_solanum <- function(size = 5, log.theta) {
    theta <- exp(log.theta)
    # parameters are in alphabetical order, it holds true for all
    # chromploid functions
    epsilon.0 <- theta[1]
    epsilon.1 <- theta[2]
    rho.0 <- theta[3]
    rho.1 <- theta[4]
    prob.01 <- theta[5]
    prob.10 <- theta[6]
    C = size
    # pure duplications
    Q <- matrix(rep(0, 4 * C * C), nrow = 2 * size)
    for (i in 1:3) {
        Q[i, (i + 2)] <- rho.0
        Q[(5 + i), (i + 7)] <- rho.1
    }
    # demiploidies
    for (i in 1:4) {
        Q[i, (i + 1)] <- epsilon.0
        Q[(5 + i), (i + 6)] <- epsilon.1
    }
    # transitions between binary state
    for (i in 1:5) {
        Q[i, (i + 5)] <- prob.01
        Q[(5 + i), i] <- prob.10
    }
    # diagonal of Q-matrix sums to zero
    diagQ <- apply(Q, 1, sum)
    for (i in 1:10) {
        Q[i, i] <- -diagQ[i]
    }
    return(Q)
}
```

To defined the Q-matrix for the reduced model where $\rho = \rho_H = \rho_W$ we do exactly the same as before but we equal the vector entry number 3 to the two parameters, that is

```r
Q_solanumred <- function(size = 5, log.theta) {
    theta <- exp(log.theta)
    epsilon.0 <- theta[1]
    epsilon.1 <- theta[2]
    rho.0 <- theta[3]
    # these two rhos have the same entry of theta which is size 5
    # instead of 6
    rho.1 <- theta[3]
    prob.01 <- theta[4]
    prob.10 <- theta[5]
    C = size
    Q <- matrix(rep(0, 4 * C * C), nrow = 2 * size)
    for (i in 1:3) {
        Q[i, (i + 2)] <- rho.0
        Q[(5 + i), (i + 7)] <- rho.1
    }
    for (i in 1:4) {
        Q[i, (i + 1)] <- epsilon.0
        Q[(5 + i), (i + 6)] <- epsilon.1
    }
    for (i in 1:5) {
        Q[i, (i + 5)] <- prob.01
        Q[(5 + i), i] <- prob.10
    }
    diagQ <- apply(Q, 1, sum)
    for (i in 1:10) {
        Q[i, i] <- -diagQ[i]
    }
    return(Q)
}
```

Those are the only two definitions needed for chromploid to perform a likelihood ratio test.

Optimization of the full and reduced models

The optimization process requires you to assign a probability distribution at the root of the tree. In this example is uniform (we really don't know the chromosome number or binary trait at the root). We also use some starting values in the optimization called `x.0`.

Here are the steps to optimize the full model

```r
mytree<-read.tree("solanumwoody2.tre")
mysample<-read.csv("solanumtransformed.csv", header=FALSE)
#this contains the dataset transformed, the orginal dataset is in solanumsimpledataset.csv
#5 states for woody and 5 for herbaceous coded 0=Herbaceous 1=woody
last.state=5

#Uniform distribution for root
p.0<-rep(1,2*(last.state))/(2*(last.state))

#Initial parameter values for optimization (random)
x.0<-log(c(0.182742343,0.214736895,0.787883643,0.550296163,0.553774817,0.368289079))
```

```r
#store results
resultsfull<-rep(0,8)
# Options used in nloptr, for more information ?nloptr. This
# is a subplex algorithm, with high tolerance to fine a very
# precise optimum.
my.options<-
list("algorithm"="NLOPT_LN_SBPLX","ftol_rel"=1e-08,"print_level"=1,
     "maxtime"=170000000, "maxeval"=5000)
#Full model optimization
full.model<- nloptr(x0 = x.0, eval_f = chromploid_nllike, opts = my.options,
    phy = mytree, tip.values = mysample, pi.0 =p.0, Q.FUN = Q_solanum,
Q.ARGS = list(size = 5))
print(full.model)
resultsfull[1:6]<-mle$solution
        resultsfull[7]<-mle$objective
        resultsfull[8]<-mle$status}
        resultsfull<-as.dataframe(resultsfull)
        names(results)<-
          c("epsilon0","epsilon1","rho0","rho1","q01","q10","nloglike","convergencestatus")
```

Now the reduced model optimization works exactly the same but replacing `x.0` for a vector of 5 parameter (because the polyploidy rates are equal), and `Q_solanum` for `Q_solanum_red`.

```r
#Initial parameter values for optimization (random)
x.0<-log(c(0.626796332,0.889894765,0.78642404,0.786166935,0.634590639))

#store results
resultsred<-rep(0,7)
# Options used in nloptr, for more information ?nloptr. This
# is a subplex algorithm, with high tolerance to fine a very
# precise optimum.
my.options<-
  list("algorithm"="NLOPT_LN_SBPLX","ftol_rel"=1e-08,"print_level"=1,
       "maxtime"=170000000, "maxeval"=5000)
#Full model optimization
reduced.model<- nloptr(x0 = x.0, eval_f = chromploid_nllike, opts = my.options,
    phy = mytree, tip.values = mysample, pi.0 =p.0, Q.FUN = Q_solanum,
Q.ARGS = list(size = 5))
print(reduced.model)
resultsred[1:5]<-mle$solution
        resultsred[6]<-mle$objective
        resultsred[7]<-mle$status}
        resultsred<-as.dataframe(resultsred)
        names(resultsred)<-
          c("epsilon0","epsilon1","rho","q01","q10","nloglike","convergencestatus")
```

Likelihood ratio test

Since you already calculated the negative log-likelihoods using `nloptr` in `chromploid_nllike` function you only need to calculate statistic `D` and the p-value as follows.

```r
neglog.red <- resultsred$nloglike
neglog.full <- resultsfull$nloglike
```

```r
alpha <- 0.05
D = 2 * (neglog.red - neglog.full)
p.value <- pchisq(D, lower.tail = FALSE, df = 1)
if (p.value < 0.05) {
    reject <- 0
} else {
    reject <- 1
}
```

Some more details

The dataset has chromosome numbers ranging from 12 to 48. To make this models work and not having to specify really large matrices with lots of zeros it is important to recode the original dataset into 10 states as follows:

Herbaceous Csomes	Woody Csomes
12=1	12=6
18=2	18=7
24=3	24=8
36=4	18=9
36=5	18=10

And those are the 10 states used by `Q_solanum` and what you would find in the transformed dataset.