Multiclade environment-dependent diversification

Jan Smycka

August 10, 2018

Generate testing dataset

I'd like to estimate common diversification parameters of environment-dependent diversification model across multiple phylogenetic trees. So let's generate a set of trees coming from a process with the same parameters.

To do this, let's first define a function for generating time-dependent trees. (This should be present in RPANDA according to the documentation, but is missing in the current version. I downloaded it from Helene Morlon's github.)

```
library(RPANDA)
```

```
## Loading required package: picante
## Loading required package: ape
## Loading required package: vegan
## Loading required package: permute
## Loading required package: lattice
## This is vegan 2.4-5
## Loading required package: nlme
```

```
library(geiger)
library(pspline)

sim_env_bd <- function (env_data, f.lamb, f.mu, lamb_par, mu_par, df=NULL, time.stop = 0, return.all.ex

{

  birthdeath.tree.timevar_simp <- function (f.lamb, f.mu, lamb_par, mu_par, time.stop = 0, return.all.e

  {
    if (time.stop == 0)
      stop("Must have stopping time")

    while (1) {
      nblineages<-c(1)
      times<-c(0)
      b<-f.lamb(0,lamb_par)
      d<-f.mu(0,mu_par)
      dt <- rexp(1,(b + d))
      t<-dt

      if (t >= time.stop)
      {
        t <- time.stop
        alive<-1
        times<-c(times,t)
```

```
    nblineages<-c(nblineages,1)
    break}

  r <- runif(1)

  if (r>b/(b + d))
  {
    times<-c(times,dt)
    nblineages<-c(nblineages,0)
    alive<-rep(FALSE,1)}

  else
  {
    edge <- rbind(c(1, 2), c(1, 3))
    edge.length <- rep(NA, 2)
    stem.depth <- rep(t, 2)
    alive <- rep(TRUE, 2)
    times<-c(times,dt)
    nblineages<-c(nblineages,sum(alive))
    next.node <- 4
    repeat
    {
      if (sum(alive) == 0)
        break
      b<-f.lamb(t,lamb_par)
      d<-f.mu(t,mu_par)
      dt <- rexp(1, sum(alive) * (b + d))
      t <- t + dt
      if (t >= time.stop)
      {
        t <- time.stop
        times<-c(times,t)
        nblineages<-c(nblineages,sum(alive))
        break}

      r <- runif(1)
      if (r <= b/(b + d))
      {
        random_lineage <- round(runif(1, min = 1, max = sum(alive)))
        e <- matrix(edge[alive, ], ncol = 2)
        parent <- e[random_lineage, 2]
        alive[alive][random_lineage] <- FALSE
        edge <- rbind(edge, c(parent, next.node), c(parent,next.node + 1))
        next.node <- next.node + 2
        alive <- c(alive, TRUE, TRUE)
        stem.depth <- c(stem.depth, t, t)
        x <- which(edge[, 2] == parent)
        edge.length[x] <- t - stem.depth[x]
        edge.length <- c(edge.length, NA, NA)
        times<-c(times,t)
        nblineages<-c(nblineages,sum(alive))}

      else
```

```
        {
          random_lineage <- round(runif(1, min = 1, max = sum(alive)))
          edge.length[alive][random_lineage] <- t - stem.depth[alive][random_lineage]
          alive[alive][random_lineage] <- FALSE
          times<-c(times,t)
          nblineages<-c(nblineages,sum(alive))}
      }
    }

    if (return.all.extinct == TRUE | sum(alive) > 0)
    {
      break}
  }



  if (sum(alive)==0) {obj<-NULL}
  else if (sum(alive)==1) {obj<-1}
  else
  {
    edge.length[alive] <- t - stem.depth[alive]
    n <- -1
    for (i in 1:max(edge)) {
      if (any(edge[, 1] == i)) {
        edge[which(edge[, 1] == i), 1] <- n
        edge[which(edge[, 2] == i), 2] <- n
        n <- n - 1}}
    edge[edge > 0] <- 1:sum(edge > 0)
    tip.label <- 1:sum(edge > 0)
    mode(edge) <- "character"
    mode(tip.label) <- "character"
    obj <- list(edge = edge, edge.length = edge.length, tip.label = tip.label)
    class(obj) <- "phylo"
    obj <- old2new.phylo(obj)
    if (prune.extinct)
    {obj<-drop.extinct(obj)}
  }
  return(list("tree"=obj,"times"=times,"nblineages"=nblineages))
}

if (is.null(df))
{
  df <- smooth.spline(x=env_data[,1], env_data[,2])$df
}
spline_result <- smooth.spline(env_data[,1],env_data[,2], df=df)
env_func <- function(t){predict(spline_result,t)}

lower_bound_control <- 0.10
upper_bound_control <- 0.10
lower_bound <- min(env_data[,1])
upper_bound <- max(env_data[,1])
time_tabulated <- seq(from=lower_bound*(1.0-lower_bound_control),
                      to=upper_bound*(1.0+upper_bound_control),
```

```
                        length.out=1+1e6)
  env_tabulated <- env_func(time_tabulated)
  env_func_tab <- function(t)
  {
    b <- upper_bound * (1.0 + upper_bound_control)
    a <- lower_bound * (1.0 - lower_bound_control)
    n <- length(env_tabulated) - 1
    index <- 1 + as.integer( (t - a) * n / (b - a))
    return(env_tabulated[index])
  }
  f.lamb.env <- function(t,y){ f.lamb(t, env_func(time.stop-t)$y, y)}
  f.mu.env <- function(t,y){ f.mu(t, env_func(time.stop-t)$y, y)}

  res <- birthdeath.tree.timevar_simp(f.lamb.env, f.mu.env, lamb_par, mu_par, time.stop, return.all.ext

  return(res)

}
```

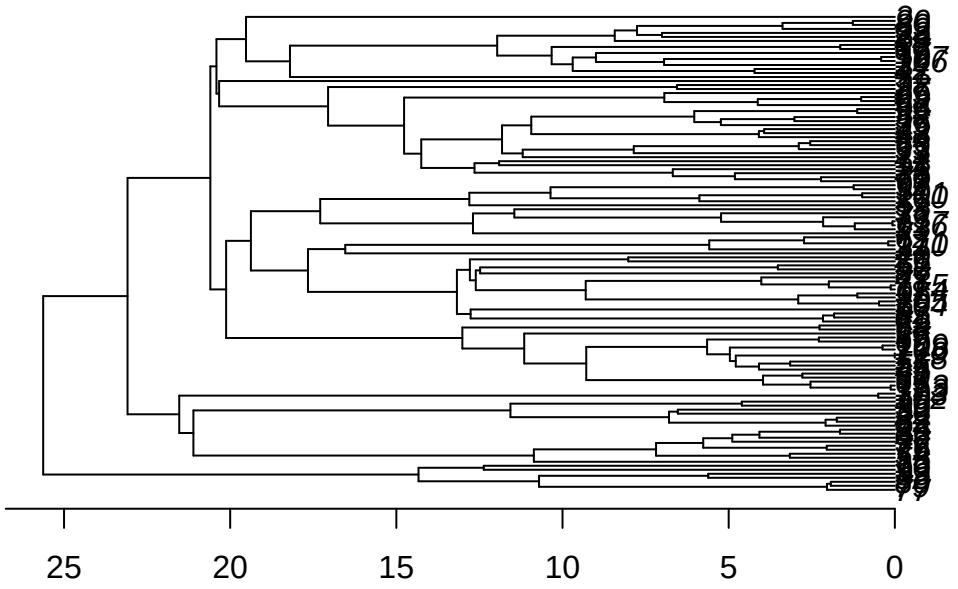
Now I can generate 20 phylogenetic trees with speciation exponentially dependent on past temperature and no extinction, with lambda0=0.1 and alpha=0.05. We constrain generation time to 30 My, so they are reasonably large, and for sake of simplicity I assume that the trees have 100% sampling.

```
data("InfTemp")

#functions for simulation
dof<-smooth.spline(InfTemp[,1], InfTemp[,2])$df
f.lamb <-function(t,x,y){y[1] * exp(y[2] * x)}
f.mu<-function(t,x,y){0}
real=c(0.1, 0.05)
mu_par<-c()
sim_envlist=function(time.stop){sim_env_bd(InfTemp,f.lamb,f.mu,real,mu_par,time.stop=time.stop)$tree}

#simulation
tlist=lapply(rep(30,20), sim_envlist)

#tree 1
plot(tlist[[1]])
axisPhylo()
```




```
#get total time for each tree
tot_timefn=function(phylo){max(node.age(phylo)$ages)}
tot_timelist=lapply(tlist,tot_timefn)

#list of sampling proportion
flist=as.list(rep(1,20))
```

Functions of multiclade environment-dependent model

Lets define the multiclade fit_bd and fit_env. The main change is in likelihood definition in fit_bd, which is a sum of likelihoods for different trees, the rest is just a wrapper. All the changes to the original functions are commented.

```
fit_bdmulti=function (phylolist, tot_timelist, f.lamb, f.mu, lamb_par, mu_par, flist,
          meth = "Nelder-Mead", cst.lamb = FALSE, cst.mu = FALSE, expo.lamb = FALSE,
          expo.mu = FALSE, fix.mu = FALSE, dt = 0, cond = "crown")
{
  #the calculation of total n for aicc
  ntiplist=lapply(phylolist,Ntip)
  nobs=Reduce("+",ntiplist)

  if (fix.mu == FALSE) {
    init <- c(lamb_par, mu_par)
    p <- length(init)
    optimLH <- function(init) {
      lamb_par <- init[1:length(lamb_par)]
      mu_par <- init[(1 + length(lamb_par)):length(init)]
      f.lamb.par <- function(t) {
        abs(f.lamb(t, lamb_par))
      }
      f.mu.par <- function(t) {
        abs(f.mu(t, mu_par))
      }
      #the LH is now a sum of likelihoods for different trees, each tree can have a different total age
      #and sampling proportion (f)
      lhsinpar=function(phylo, tot_time, f){likelihood_bd(phylo, tot_time, f.lamb.par,
                                            f.mu.par, f, cst.lamb = cst.lamb, cst.mu = cst.mu,
                                            expo.lamb = expo.lamb, expo.mu = expo.mu, dt = dt,
                                            cond = cond)}
      likelihoodlist=mapply(lhsinpar, phylolist, tot_timelist, flist)
      LH=Reduce("+",likelihoodlist)

      return(-LH)
    }
    temp <- suppressWarnings(optim(init, optimLH, method = meth))
    lamb.par <- temp$par[1:length(lamb_par)]
    mu.par <- temp$par[(1 + length(lamb_par)):length(init)]
    f.lamb.par <- function(t) {
      f.lamb(t, lamb.par)
    }
    f.mu.par <- function(t) {
      f.mu(t, mu.par)
    }
    res <- list(model = "birth death", LH = -temp$value,
                aicc = 2 * temp$value + 2 * p + (2 * p * (p + 1))/(nobs -
                                                                     p - 1), lamb_par = lamb.par, mu_pa
                f.lamb = Vectorize(f.lamb.par), f.mu = Vectorize(f.mu.par))
  }
  else {
    init <- c(lamb_par)
    p <- length(init)
```

```
    optimLH <- function(init) {
      lamb_par <- init[1:length(lamb_par)]
      f.lamb.par <- function(t) {
        abs(f.lamb(t, lamb_par))
      }
      f.mu.par <- function(t) {
        abs(f.mu(t, mu_par))
      }
      #the LH is now a sum of likelihoods for different trees, each tree can have a different total age
      #and sampling proportion (f)
      lhsinpar=function(phylo, tot_time, f){likelihood_bd(phylo, tot_time, f.lamb.par,
                                                          f.mu.par, f, cst.lamb = cst.lamb, cst.mu = cst.mu
                                                          expo.lamb = expo.lamb, expo.mu = expo.mu, dt = dt
                                                          cond = cond)}
      likelihoodlist=mapply(lhsinpar, phylolist, tot_timelist, flist)
      LH=Reduce("+",likelihoodlist)

      return(-LH)
    }
    temp <- suppressWarnings(optim(init, optimLH, method = meth))
    lamb.par <- temp$par[1:length(lamb_par)]
    f.lamb.par <- function(t) {
      f.lamb(t, lamb.par)
    }
    f.mu.par <- function(t) {
      f.mu(t, mu_par)
    }
    res <- list(model = "birth.death", LH = -temp$value,
                aicc = 2 * temp$value + 2 * p + (2 * p * (p + 1))/(nobs -
                                                                     p - 1), lamb_par = lamb.par, f.lamb
  }
  class(res) <- "fit.bd"
  return(res)
}
```

```
fit_envmulti=function (phylolist, env_data, tot_timelist, f.lamb, f.mu, lamb_par,
          mu_par, df = NULL, flist, meth = "Nelder-Mead", cst.lamb = FALSE,
          cst.mu = FALSE, expo.lamb = FALSE, expo.mu = FALSE, fix.mu = FALSE,
          dt = 0, cond = "crown")
{
  if (is.null(df)) {
    df <- smooth.spline(x = env_data[, 1], env_data[, 2])$df
  }
  spline_result <- sm.spline(env_data[, 1], env_data[, 2],
                             df = df)
  env_func <- function(t) {
    predict(spline_result, t)
  }
  lower_bound_control <- 0.1
  upper_bound_control <- 0.1
  lower_bound <- min(env_data[, 1])
  upper_bound <- max(env_data[, 1])
  time_tabulated <- seq(from = lower_bound * (1 - lower_bound_control),
                        to = upper_bound * (1 + upper_bound_control), length.out = 1 +
                          1e+06)
  env_tabulated <- env_func(time_tabulated)
  env_func_tab <- function(t) {
    b <- upper_bound * (1 + upper_bound_control)
    a <- lower_bound * (1 - lower_bound_control)
    n <- length(env_tabulated) - 1
    index <- 1 + as.integer((t - a) * n/(b - a))
    return(env_tabulated[index])
  }
  f.lamb.env <- function(t, y) {
    f.lamb(t, env_func_tab(t), y)
  }
  f.mu.env <- function(t, y) {
    f.mu(t, env_func_tab(t), y)
  }
  #here we use fit_bdmulti instead of fit_bd
  res <- fit_bdmulti(phylolist, tot_timelist, f.lamb.env, f.mu.env, lamb_par,
                mu_par, flist, meth=meth, cst.lamb, cst.mu, expo.lamb, expo.mu,
                fix.mu, dt, cond)
  res$model <- "environmental birth death"
  res$f.lamb <- function(t) {
    f.lamb(t, env_func_tab(t), res$lamb_par)
  }
  if (fix.mu == FALSE) {
    res$f.mu <- function(t) {
      f.mu(t, env_func_tab(t), res$mu_par)
    }
  }
  class(res) <- "fit.env"
  return(res)
}
```

Fitting the multiclade model

OK, so now I'm ready to fit the multiclade model.

```
#starting parameters for optimizer
lamb_par<-c(0.2, 0.2)
mu_par<-c()

#fitting
artifmulti=fit_envmulti(tlist,InfTemp,tot_timelist,f.lamb=f.lamb,f.mu=f.mu,lamb_par=lamb_par,mu_par=mu_
artifmulti
```

```
## model :
## [1] "environmental birth death"
##
## LH :
## [1] -3485.989
##
## aicc :
## [1] 6975.987
##
## lamb_par :
## [1] 0.09832543 0.05515760
```

Fitting a model for each clade separately

For comparison, I also fit each tree separately.

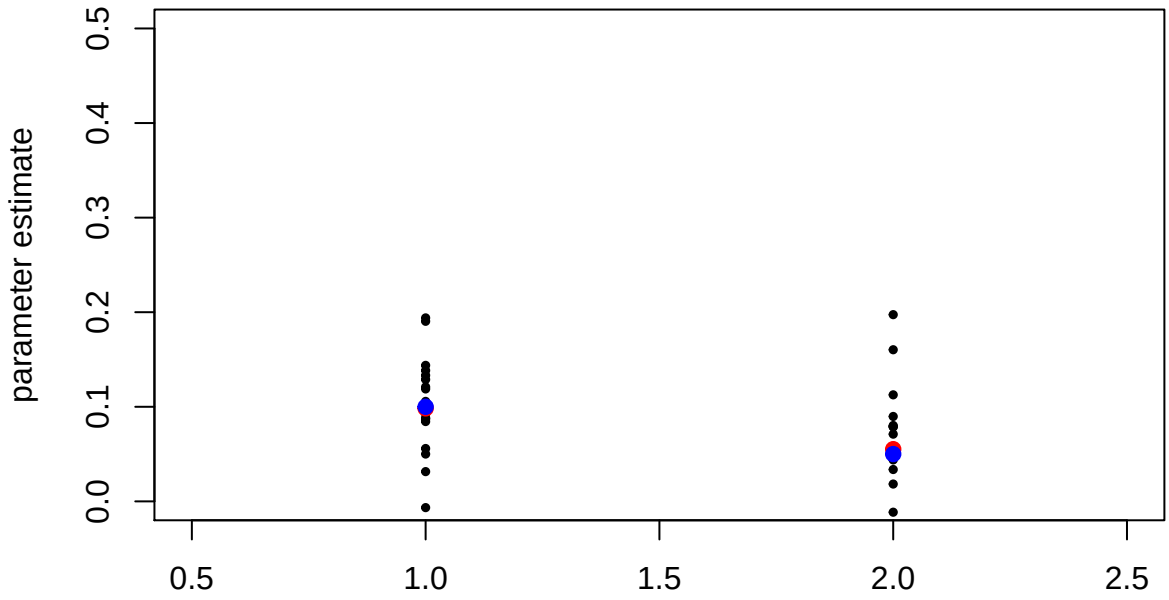
```
fered=function(phylo, tot_time){fit_env(phylo,InfTemp,tot_time,f.lamb=f.lamb,f.mu=f.mu,lamb_par=lamb_pa
artiflist=mapply(fered,tlist,tot_timelist)

#vectorize model characteristics froma list
lamb_par1=rep(0,20)
lamb_par2=rep(0,20)
aicclist=rep(0,20)
LHlist=rep(0,20)
for(i in 1:20){
  tempres=artiflist[,i]
  lamb_par1[i]=tempres$lamb_par[1]
  lamb_par2[i]=tempres$lamb_par[2]
  aicclist[i]=tempres$aicc
  LHlist[i]=tempres$LH
}
```

Estimates comparison

Now I can compare parameter estimates of single-clade (small black) and multi-clade (big red) models with generating values (big blue). First parameter is lambda0, second is alpha.

```
plot(c(lamb_par1,lamb_par2)~c(rep(1,20),rep(2,20)),xlim=c(0.5,2.5),ylim=c(0,0.5),xlab="",ylab="paramete
points(1,artifmulti$lamb_par[1], col="red", pch=19)
points(2,artifmulti$lamb_par[2], col="red", pch=19)
points(1,real[1], col="blue", pch=19)
points(2,real[2], col="blue", pch=19)
```



AIC comparison

And I can also compare the AICs of the models. The AIC of the multi-clade model can be directly calculated from its loglikelihood.

```
aictog=-2*(artifmulti$LH-length(lamb_par))
```

The AIC of single-clade model fits can be obtained based on fact that optimizing multiple functions separately yields the same results as optimizing a sum of them, each having different parameters. In other words, the set of single-clade models is equivalent to a model where each tree is fitted with its own parameter. Based on this, the AICs can be calculated from sum of loglikelihoods and model parameters.

Note: I didn't use the AICc, cause I weren't sure what if I can calculate number of observations just by summing up all the tips.

```
aicsep=-2*(sum(LHlist)-length(lamb_par)*20)
```

The multi-clade model has lower AIC. This makes sense as all the clades were indeed generated by the same process. (Here one should to do some sensitivity analysis, but it should work as any other deltaAIC comparison.)

```
aictog
```

```
## [1] 6975.977
```

```
aicsep
```

```
## [1] 7018.224
```

Task for future

What would be really cool is putting some other "mixed effect model" features in place. For example fixing a common alpha parameter for all the clades, while leaving the lambda0 clade-specific. I.e. all the clades react to the environment in the came way, but they have different baseline rate of speciation.