Fel Pruning Ill Condition

Pruning Algorithm vs Ill conditioning Tree

There is an important overlook of the Felsenstein's pruning algorithm here. One of the advantages of this method is to make sure that the computation of the likelihood can be done properly.

> **Question**: Does the application of the pruning algorithm completely solves the problem even when G matrices are ill-conditioned?
>
> **Ans:** No! Pruning Algorithm fails for some ill-conditioned G matrices. The following gives an example using 3 taxa case.

Pruning Algorithm

We use `mvMorph` function `pruning(tree)` which applies the pruning algorithm (Felsenstein 1973) to efficiently compute the square root of the matrix (or its inverse) C.

Given tree and its C matrix, the statistical model of trait evolution of Y assuming Brownian motion is

$$Y \sim N(\mu \mathbf{1}, \sigma^2 C).$$

According to the function `pruning`, the inverse square root of the covariance $C^{-1/2}$ is calculated based on the pruning algorithm. Then the trait Y is transformed into Z as

$$Z = C^{-1/2}Y \sim N(\mu C^{-1/2}\mathbf{1}, \sigma^2 \mathbf{I})$$

Now the observation is indepdent and normally distrbuted. The negative log likelihood function is

$$\ell(\mu, \sigma^2 | Z, C) = \frac{n}{2}\log 2\pi + \frac{n}{2}\log \sigma^2 + \frac{1}{2\sigma^2}(Z - \mu C^{-1/2}\mathbf{1})^t (Z - \mu C^{-1/2}\mathbf{1})$$

Taking partial derivative with respect to μ and σ^2

$$\frac{\partial \ell}{\partial \mu} = 0, \frac{\partial \ell}{\partial \sigma^2} = 0$$

The MLEs are

$$\hat{\mu} = \frac{Z^t C^{-1/2}\mathbf{1}}{(C^{-1/2}\mathbf{1})^t (C^{-1/2}\mathbf{1})}, \hat{\sigma}^2 = \frac{(Z - \hat{\mu} C^{-1/2}\mathbf{1})^t (Z - \hat{\mu} C^{-1/2}\mathbf{1})}{n}.$$

In the following code with 3 taxa example, current with a tree who conditione number is $\approx 10^6$, the tree pruning algorithm fails to estimate the parameters.

```
set.seed(13912)
rm(list=ls())
library(mvMORPH)
```

```
## Loading required package: phytools
```

```
## Loading required package: ape
```

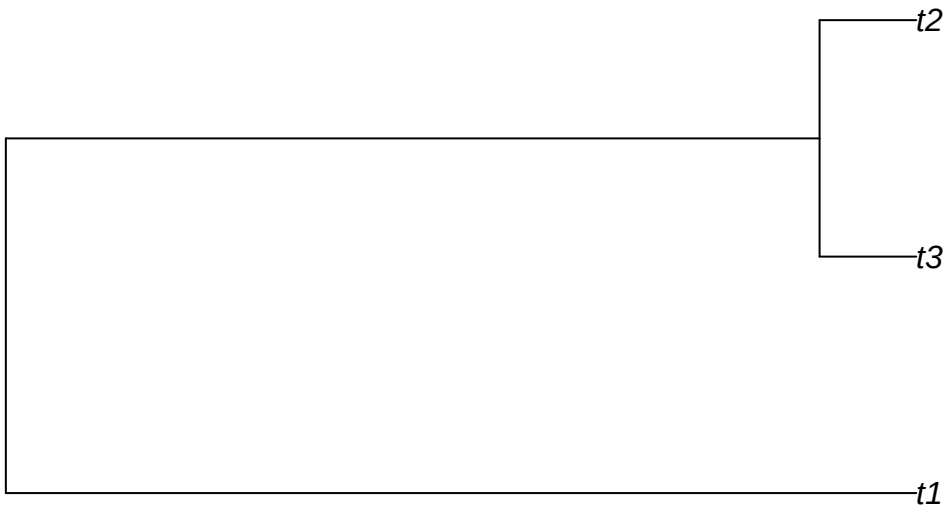
```
## Loading required package: maps

## Loading required package: corpcor

## Loading required package: subplex

## ##
## ## mvMORPH package (1.1.0)

## ## Multivariate evolutionary models

## ##
## ## See the tutorials: browseVignettes("mvMORPH")

## ##
## ## To cite package 'mvMORPH': citation("mvMORPH")
## ##
```

```
library(phytools)
library(phyclust)
library(ape)
library(TreeSim)
```

```
## Loading required package: geiger

## Registered S3 method overwritten by 'geiger':
##   method            from
##   unique.multiPhylo ape

##
## Attaching package: 'geiger'

## The following object is masked from 'package:mvMORPH':
##
##     aicw
```

```
library(geiger)
library(phangorn)
library(MASS)

# NEED to check tinytipvcv MAKE THIS MATRI POSTIVE DEFINTE
tinytipvcv<-function(C=C,shrink=shrink){
  C<-C/max(C)
  uniC<-unique(c(C))
  secondlargest<-  sort(uniC)[(length(uniC)-1)]
  diag(C)<-(max(C)-secondlargest)*shrink+secondlargest
  return(C)
}

svd_decom<-function(M){#returns M^{-1}.
  decom<-svd(M)
  val<-decom$d
  solveD<-diag(1/val)
  return(decom$v%*%solveD%*%t(decom$u))
}

mu<-0
sigmasq<-1

treesize<- 3
```

```
tol <- 1e-30

tree<-sim.bd.taxa.age(n=treesize,numbsim=1,lambda=1,mu=0.5,age=treesize,mrca=TRUE)[[1]]
plot(tree)
```



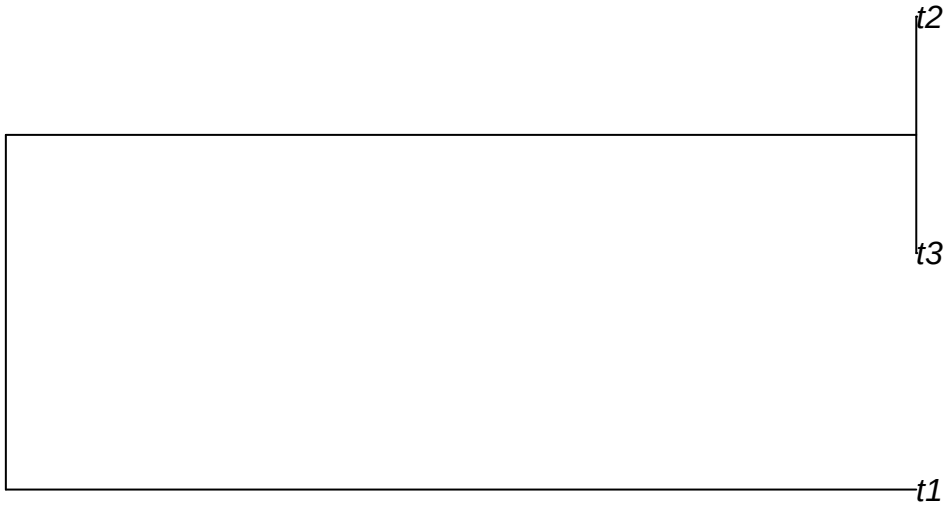

```
## HERE WE TEST WHETHER PRUNING WORK FOR KAPPA = Inf

Ctiny <-  tinytipvcv(C=vcv(tree),shrink=tol) # Bad Tree we use a very bad tree to simulate data, as exp
kappa(Ctiny)
```

```
## [1] 1.138549e+16
```

```
eigen(Ctiny)$values
```

```
## [1] 1.787627e+00 8.938134e-01 1.110223e-15
```

```
D<- 2*(diag(Ctiny)[1]-Ctiny)
treeupgma<-upgma(D)
plot(treeupgma)
```



Here the condition number is

```
kappa(vcv(treeupgma))
```

```
## [1] 1.138549e+16
```

And we check the $C^{-1}C = I$ for four method `solve`, `ginv`, `svd_decom`

```
try(round(solve(vcv(treeupgma))%*%vcv(treeupgma),4))
```

```
## Error in solve.default(vcv(treeupgma)) :
##   Lapack routine dgesv: system is exactly singular: U[3,3] = 0
```

```
try(round(ginv(vcv(treeupgma))%*%vcv(treeupgma),4))
```

```
##      t1  t3  t2
## [1,]  1 0.0 0.0
## [2,]  0 0.5 0.5
## [3,]  0 0.5 0.5
```

```
try(round(svd_decom(vcv(treeupgma))%*%vcv(treeupgma),4))
```

```
##       t1  t3  t2
## [1,] NaN NaN NaN
## [2,] NaN NaN NaN
## [3,] NaN NaN NaN
```

check the pruning algorithm work well with the tree structure

```
t(pruning(treeupgma,inv = FALSE)$sqrtMat) %*% pruning(treeupgma,inv = FALSE)$sqrtMat
```

```
##      [,1] [,2] [,3]
## [1,]  NaN  NaN  NaN
## [2,]  NaN  NaN  NaN
## [3,]  NaN  NaN  NaN
```

```
vcv(treeupgma)
```

```
##           t1        t3        t2
## t1 0.8938134 0.0000000 0.0000000
## t3 0.0000000 0.8938134 0.8938134
## t2 0.0000000 0.8938134 0.8938134
```

Here the **pruning algorithm** fails

```
C0.5treeupgma <-  pruning (treeupgma,inv = FALSE)
Cinv0.5treeupgma<-pruning(treeupgma,inv = TRUE)
Cinv0.5treeupgma
```

```
## $sqrtMat
##      [,1] [,2] [,3]
## [1,]  NaN  Inf -Inf
## [2,]  NaN  NaN  NaN
## [3,]  NaN  NaN  NaN
##
## $varNode
## [1] NaN   0
##
## $varRoot
## [1] NaN
##
## $det
## [1] NaN
##
```

```
## attr(,"class")
## [1] "mvmorph.var"
```

Hence the parameter estimation fails

```
one<-array(1,c(dim(Ctiny)[1],1))

y<- matrix(mvrnorm(n = 1, mu*one, Sigma=sigmasq*vcv(treeupgma), tol = 1e-6, empirical = FALSE, EISPACK =
y
```

```
##             [,1]
## [1,] -0.4063844
## [2,] -0.1410536
## [3,] -0.1410536
```

```
Cinv0.5treeupgma.sqrtMat<-Cinv0.5treeupgma$sqrtMat
z<-Cinv0.5treeupgma.sqrtMat%*%y
z
```

```
##      [,1]
## [1,]  NaN
## [2,]  NaN
## [3,]  NaN
```

```
new.one <- Cinv0.5treeupgma.sqrtMat%*%one
mu.hat <-  c((t(z)%*%new.one)  /    (t(new.one)%*%new.one))
sigmasq.hat<- t(z-mu.hat*Cinv0.5treeupgma.sqrtMat%*%one)%*%(z-mu.hat*Cinv0.5treeupgma.sqrtMat%*%one)/le
c(mu.hat,sigmasq.hat)
```

```
## [1] NaN NaN
```