A Tutorial for the patherit package

Venelin Mitov

Introduction

The **patherit** package provides an implementation of the Pylogenetic Mixed Model assuming an Ornstein-Uhlenbeck (OU) or a Brownian Motion (BM) process of trait evolution along the branches. The package comes with an efficient likelihood calculation that scales to large phylogenies (currently tested with up to 100000 tips) and, if needed, can perform the likelihood calculation using the package **Rmpfr** for high precision floating point operations. Currently, the package depends on the packages **ape**, **data.table**, **coda** and **adaptMCMC**. The most important functions of the package are "lik.poumm" (see ?lik.poimm), which performs likelihood calculation, "ml.poumm" (type ?ml.poumm), which fits POUMM to a tree and data at the tips using maximum likelihood and mcmc.poumm, which runs an MCMC fit to a tree and data. The algorithm for likelihood calculation has been extensively tested and applied on various real as well as synthetic data.

Note: The high-precision likelihood calculation is usually needed in case of accumulating numerical errors due to extreme values of some of the parameters alpha, theta, sigma or sigmae or extreme (i.e. very long or very short) branch lengths in the phylogeny. As the Rmpfr package is currently not supported on all linux platforms, the poumm package doesn't explicitly depend on it, but you will need to install it manually and import it in case you specify usempfr=2 in the call to lik.poumm, ml.poumm or mcmc.poumm.

Installing the package

```r
install.packages('patherit_0.2.tgz', type='binary', repos=NULL)
```

A Hello World example

```r
library(patherit)

set.seed(1)
# Number of tips
N <- 1000
# true parameters
alpha <- 1
theta <- 4.5
sigma <- 2
sigmae <- 2

tree <- rtree(N)
tree$edge.length <- tree$edge.length/10

g <- generateTraitOU(tree, g0=6, alpha=alpha, theta=theta, sigma=sigma)
z <- rnorm(N, mean=0, sd=sigmae)+g[1:N]
```

One likelihood calculation

```
# you need to do this only for performance speed-up and if you have
# multiple explicit calls to lik.poumm
pruneInfo <- pruneTree(tree)
#likelihood at the true value
lik <- lik.poumm(z, tree, alpha, theta, sigma, sigmae, log=T, distgr='maxlik', pruneInfo=pruneInfo)
```

Performing an ML-fit

```
# maximum likelihood fit
mlfit <- ml.poumm(z=z, tree=tree, distgr='maxlik', parMax=c(alpha=10, theta=10, sigma=10, sigmae=10))

print(mlfit$par)
```

```
##    alpha    theta    sigma   sigmae
## 0.723194 6.900631 1.624662 1.993503
```

```
# ml likelihood value versus value at the original true params
print(c(-mlfit$value, lik))
```

```
## [1] -2181.628 -2187.276
```

Performing an MCMC fit

```
mcmcfit <- mcmc.poumm(z, tree, n.mcmc=2e5, n.adapt=20000, thin=100, acc.rate=0.1, divideEdgesBy=100,
                      scale=matrix(c(400,   0.00,  0.00,  0.00,
                                     0.00,  2.00,  0.00,  0.00,
                                     0.00,  0.00,  0.02,  0.00,
                                     0.00,  0.00,  0.00,  0.02), nrow=4, ncol=4, byrow=T),
                      distgr="maxlik")
```

```
## Warning in window.mcmc(mc$mcmc, start = start, end = end, thin = thin): end
## value not changed
```

```
## Warning in FUN(X[[i]], ...): end value not changed
```

```
# this produces a worning which should be harmless
mcmcAnalysis <- analyseMCMCs(chains=mcmcfit$chains,
                             stat=function(par) {
                               H2e.poumm(z=z, sigmae=sqrt(par[4]))
                             },
                             statName='H2.OUe',
                             start=1e5, end=2e5, thin=100)
```

```
## Warning in window.mcmc(mc$mcmc, start = start, end = end, thin = thin): end
## value not changed
```

```
## Warning in FUN(X[[i]], ...): end value not changed
```

Comparing point estimates

```
# point-estimates of broad-sense heritability.
# A difference between H2eq and H2e, i.e. H2eq>H2e shows that the process has'nt still
# converged to the stationary OU distribution.
print(c(
  H2eq.true=H2.poumm(alpha, sigma, sigmae, t=Inf, tm=0),
  H2eq.ml.poumm=H2.poumm(mlfit$par[1], mlfit$par[3], mlfit$par[4], t=Inf, tm=0),
  H2e.ml.poumm=H2e.poumm(z=z, sigmae=mlfit$par[4]),
  H2e.mcmc.poumm=mcmcAnalysis$Mode))
```

```
##           H2eq.true       H2eq.ml.poumm H2e.ml.poumm.sigmae
##           0.3333333           0.3146958           0.2131682
##      H2e.mcmc.poumm
##           0.2077703
```

```
# 95% CI from the MCMC
print(mcmcAnalysis$HPD)
```

```
## [[1]]
##         lower     upper
## var1 0.115609 0.2925019
## attr(,"Probability")
## [1] 0.95
```

Further information and help

For further information, look in the package help-pages. If you cannot find the answer, please write to venelin.mitov@bsse.ethz.ch.