dcAncestralML

October 24, 2014

`dcAncestralML` *Function to reconstruct ancestral discrete states using fast maximum likelihood algorithm*

Description

`dcAncestralML` is supposed to reconstruct ancestral discrete states using fast maximum likelihood algorithm. It takes inputs both the phylo-formatted tree and discrete states in the tips. The algorithm assumes that state changes can be described by a probablistic reversible model. It first determines transition matrix between states (also considering branch lengths), then uses dynamic programming (from tips to the root) to estimate conditional maximum likelihood, and finally reconstructs the ancestral states (from the root to tips). If the ties occur at the root, the state at the root is set to the last state in ties (for example, usually being 'present' for 'present'-'absent' two states).

Usage

```
dcAncestralML(x, phy, transition.model = c("different", "symmetric",
"same",
"customised"), customised.model = NULL, edge.length.power = 1,
initial.estimate = 0.1, verbose = T)
```

Arguments

`x`	a vector of discrete states in the tips. It can be an unnamed vector; in this case, assumedly it has the same order as in the tree tips. More wisely, it is a named vector, whose names can be matched to the tip labels of the tree. The names of this input vector can be more than found in the tree labels, and they should contain all those in the tree labels
`phy`	an object of class 'phylo'
`transition.model`	a character specifying the transition model. It can be: "different" for all-transition-different model (such as $matrix(c(0,1,2,0),2)$), "symmetric" for the symmetric model (such as $matrix(c(0,1,1,0),2)$ or $matrix(c(0,1,2,1,0,3,2,3,0),3)$), "same" for all-transition-same model (such as $matrix(c(0,1,1,0),2)$), "customised" for the user-customised model (see the next parameter)
`customised.model`	a matrix customised for the transition model. It can be: $matrix(c(0,1,1,0),2)$, $matrix(c(0,1,2,0),2)$, or $matrix(c(0,1,2,1,0,3,2,3,0),3)$

`edge.length.power`
: a non-negative value giving the exponent transformation of the branch lengths. It is useful when determining transition matrix between states

`initial.estimate`
: the initial value used for the maximum likelihood estimation

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

a list of architectures, containing three components for "transition", "states" and "relative":

- `transition`: an estimated transition matrix between states
- `states`: a named vector storing states (extant and ancestral states)
- `relative`: a matrix of nodes X states, storing conditional maximum likelihood being relative to each state

Note

This fast dynamic programming for ancestral discrete state reconstruction is partially inspired by a joint estimation procedure as described in http://mbe.oxfordjournals.org/content/17/6/890.full

See Also

dcAncestralMP

Examples

```
# provide the tree and states in the tips
tree <-
"((((t10:5,t2:5):2,(t9:4,t5:4):3):2,(t3:4,t7:4):6):2,((t6:4,t1:4):2,(t8:2,t4:2):4):6);"
phy <- ape::read.tree(text=paste(tree, collapse=""))
x <- c(0, rep(1,4), rep(0,5))

# reconstruct ancestral states
res <- dcAncestralML(x, phy)
res

# visualise the tree with ancestral states and their conditional probability
Ntip <- ape::Ntip(phy)
Nnode <- ape::Nnode(phy)
color <- c("white","gray")
## visualise main tree
ape::plot.phylo(phy, type="p", use.edge.length=TRUE, label.offset=1,
show.tip.label=TRUE, show.node.label=FALSE)
## visualise tips (state 1 in gray, state 0 in white)
ape::tiplabels(pch=22, bg=color[as.numeric(x)+1], cex=2, adj=1)
## visualise internal nodes
### thermo bar to illustrate conditional maximum likelihood (state 1 in gray, state 0 in white)
ape::nodelabels(thermo=res$relative[Ntip+1:Nnode,2:1],
piecol=color[2:1], cex=0.75)
### labeling reconstructed ancestral states
ape::nodelabels(text=res$states[Ntip+1:Nnode], node=Ntip+1:Nnode,
frame="none", col="red", bg="transparent", cex=0.75)
```