dcAncestralML

December 3, 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(data, phy, transition.model = c("different", "symmetric",
"same", "customised"), customised.model = NULL, edge.length.power = 1,
initial.estimate = 0.1, output.detail = F, parallel = T,
multicores = NULL, verbose = T)
```

Arguments

`data`	an input data matrix storing discrete states for tips (in rows) X characters (in columns). The rows in the matrix are for tips. If the row names do not exist, then addumedly they have the same order as in the tree tips. More wisely, users provide row names which can be matched to the tip labels of the tree. The row names 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

`output.detail`
: logical to indicate whether the output is returned as a detailed list. If TRUE, a nested list is returned: a list of characters (corresponding to columns of input data matrix), in which each element is a list consisting of three components ("states", "transition" and "relative"). If FALSE, a matrix is returned: the columns respond to the input data columns, and rows responding to all node index in the phylo-formatted tree

`parallel`
: logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`multicores`
: an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

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

Value

It depends on the 'output.detail'. If FALSE (by default), a matrix is returned, with the columns responding to the input data columns, and rows responding to node index in the phylo-formatted tree. If TRUE, a nested list is returned. Outer-most list is for characters (corresponding to columns of input data matrix), in which each elemenl is a list (inner-most) consisting of three components ("states", "transition" and "relative"):

- `states`: a named vector storing states (extant and ancestral states)
- `transition`: an estimated transition matrix between 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, dcDuplicated

Examples

```
# 1) a newick tree that is imported as a phylo-formatted tree
tree <- "(((t1:5,t2:5):2,(t3:4,t4:4):3):2,(t5:4,t6:4):6);"
phy <- ape::read.tree(text=tree)

# 2) an input data matrix storing discrete states for tips (in rows) X four characters (in columns)
data1 <- matrix(c(0,rep(1,3),rep(0,2)), ncol=1)
data2 <- matrix(c(rep(0,4),rep(1,2)), ncol=1)
data <- cbind(data1, data1, data1, data2)
colnames(data) <- c("C1", "C2", "C3", "C4")
## reconstruct ancestral states, without detailed output
res <- dcAncestralML(data, phy, parallel=FALSE)
res

# 3) an input data matrix storing discrete states for tips (in rows) X only one character
data <- matrix(c(0,rep(1,3),rep(0,2)), ncol=1)
## reconstruct ancestral states, with detailed output
res <- dcAncestralML(data, phy, parallel=FALSE, output.detail=TRUE)
res
## get the inner-most list
res <- res[[1]]
## 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)
x <- data[,1]
ape::tiplabels(pch=22, bg=color[as.numeric(x)+1], cex=2, adj=1)
## visualise internal nodes
### thermo bar to illustrate relative probability (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)
```