

# Package ‘treenomial’

February 13, 2020

**Type** Package

**Title** Comparison of Trees using a Tree Defining Polynomial

**Version** 1.1.2

**Maintainer** Matthew Gould <mgould@sfu.ca>

**Description** Provides functionality for creation and comparison of polynomials that uniquely describe trees as introduced in Liu (2019, <arXiv:1904.03332>). The core method converts rooted unlabeled phylo objects from 'ape' to the tree defining polynomials described with coefficient matrices. Additionally, a conversion for rooted binary trees with binary trait labels is also provided. Once the polynomials of trees are calculated there are functions to calculate distances, distance matrices and plot different distance trees from a target tree. Manipulation and conversion to the tree defining polynomials is implemented in C++ with 'Rcpp' and 'RcppArmadillo'. Furthermore, parallel programming with 'RcppThread' is used to improve performance converting to polynomials and calculating distances.

**Depends** R (>= 3.5.0)

**License** GPL (>= 2)

**URL** <https://github.com/mattgould/treenomial>

**Encoding** UTF-8

**Imports** Rcpp (>= 1.0.1), ape, methods

**LinkingTo** Rcpp, RcppArmadillo, RcppThread

**RoxygenNote** 7.0.2

**Suggests** knitr, rmarkdown, testthat

**NeedsCompilation** yes

**Author** Matthew Gould [aut, cre],  
Pengyu Liu [ctb],  
Caroline Colijn [ctb]

**Repository** CRAN

**Date/Publication** 2020-02-13 06:20:02 UTC

**R topics documented:**

alignPoly . . . . .	2
allTrees . . . . .	3
plotExtremeTrees . . . . .	4
polyDist . . . . .	6
polyToDistMat . . . . .	7
treeDist . . . . .	8
treeJuliaSet . . . . .	9
treeToDistMat . . . . .	10
treeToPoly . . . . .	12
wedge . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

alignPoly	<i>Align various types of coefficient matrices</i>
-----------	--

---

**Description**

Align various types of coefficient matrices

**Usage**

```
alignPoly(coefficientMatrices)
```

**Arguments**

coefficientMatrices  
a list of coefficient matrices of various sizes

**Details**

Alignment depends on the type of coefficient matrix:

“**real**” the smaller matrices columns are prepended with zero columns to align with the max number of columns and the rows are appended with zero rows to match the max number of rows

“**yEvaluated**” the smaller vectors are appended with zeroes to match the max length vector

“**tipLabel**” the smaller matrices are appended with zeroes to match the max number of rows and columns

**Value**

the aligned list of coefficient matrices

## Examples

```
library(treenomial)
library(ape)
differentSizeTrees <- c(rtree(2), rmtree(10,10))
coeffs <- treeToPoly(differentSizeTrees, numThreads = 0)
alignedCoeffs <- alignPoly(coeffs)
```

---

allTrees	<i>Calculate all full unordered m-ary trees up to n tips</i>
----------	--

---

## Description

Return normal coefficient matrices, substituted y coefficient vectors, or phylo objects for all possible unordered full m-ary trees up to n tips. For binary trees ( $m = 2$ ), the number of trees at each number of tips follows the [Wedderburn-Etherington numbers](#).

## Usage

```
allTrees(n, m = 2, type = c("default", "yEvaluated", "phylo"), y)
```

## Arguments

n	max number of tips
m	max number of children for each node
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“phylo”</b> phylo objects
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise

## Value

list of lists containing all the trees in **type** format for each number of tips

## Note

only  $m = 2$  is currently supported

**Examples**

```

library(treenomial)
library(ape)

# generate coefficient matrices describing the polynomials of all possible
# unordered full binary trees up to 10 tips

allBinTenRealCoeff <- allTrees(10, type = "phylo")

# number of trees at each number of tips follows Wedderburn-Etherington numbers
lengths(allBinTenRealCoeff)

# phylo type example, plot all 6 tip unordered full binary trees

# backup par options
oldpar <- par(no.readonly =TRUE)

allBinSixPhylo <- allTrees(6, type = "phylo")[[6]]
par(mfrow=c(1,6))
plots <- lapply(allBinSixPhylo, function(t){
  plot.phylo(ladderize(t), direction = "downwards", show.tip.label = FALSE)
})

# restore par options
par(oldpar)

```

---

plotExtremeTrees

*Plot the min/max distance trees from a target tree*


---

**Description**

Plot the min/max distance trees from a target tree

**Usage**

```

plotExtremeTrees(
  target,
  trees,
  n,
  comparison = "min",
  method = c("logDiff", "wLogDiff", "pa", "ap"),
  type = c("default", "yEvaluated", "tipLabel"),
  y,
  numThreads = -1
)

```

**Arguments**

target	the phylo object of the tree to calculate the distances to
trees	a list of phylo objects to compare with the <b>target</b>
n	the number of trees to find and plot
comparison	whether to find the “min” or the “max” distance trees from the <b>target</b>
method	method to use when calculating coefficient distances: <b>“logDiff”</b> for two coefficient matrices A and B returns $\text{sum}(\log(1+\text{abs}(A-B)))$ <b>“wLogDiff”</b> performs the “logDiff” method with weights on the rows <b>“pa”</b> total pairs where the coefficient is present in one matrix and absent in the other (presence-absence) <b>“ap”</b> opposite comparison of pa (absence-presence)
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“tipLabel”</b> complex coefficient polynomial that utilize binary trait tip labels on the phylo objects
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

**Value**

a list of lists containing the **n** min/max distance trees and their distances to **target**

**Note**

- the substituted y coefficient vector and the complex tip label coefficient matrix only support the “logDiff” method
- “pa” and “ap” force symmetry in the output distance matrix

**Examples**

```
library(treenomial)
library(ape)
trees <- c(rmtree(1000, 50), rmtree(10, 9))
target <- rtree(50)
minTrees <- plotExtremeTrees(target, trees, 2, comparison = "min", numThreads = 0)
```

---

polyDist                      *Calculates the distance between coefficient matrices*

---

### Description

Calculates the distance between two coefficient matrices or a coefficient matrix and a list of coefficient matrices.

### Usage

```
polyDist(x, Y, method = c("logDiff", "wLogDiff", "pa", "ap"), numThreads = -1)
```

### Arguments

x	single coefficient matrix to find distances to
Y	a list of coefficient matrices
method	method to use when calculating coefficient distances: <b>“logDiff”</b> for two coefficient matrices A and B returns $\sum(\log(1+\text{abs}(A-B)))$ <b>“wLogDiff”</b> performs the “logDiff” method with weights on the rows <b>“pa”</b> total pairs where the coefficient is present in one matrix and absent in the other (presence-absence) <b>“ap”</b> opposite comparison of pa (absence-presence)
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

### Value

vector of distances

### Note

- the substituted y coefficient vector and the complex tip label coefficient matrix only support the “logDiff” method
- “pa” and “ap” force symmetry in the output distance matrix

### Examples

```
library(treenomial)
library(ape)

# distance between coefficient matrices of one 10 tip tree
# and 100 trees with 30 tips using
# create the coefficient matrices
tenTipTree <- rtree(10)
tenTipTreeCoeff <- treeToPoly(tenTipTree, numThreads = 0)
```

```

thirtyTipList <- rmtree(100, 30)
thirtyTipCoeffs <- treeToPoly(thirtyTipList, numThreads = 0)

# find the distance
polyDist(tenTipTreeCoeff, thirtyTipCoeffs, numThreads = 0)

```

---

polyToDistMat

*Calculates the distance matrix from a list coefficient matrices*


---

### Description

Calculates the distance matrix from a list coefficient matrices

### Usage

```

polyToDistMat(
  coefficientMatrices,
  method = c("logDiff", "wLogDiff", "pa", "ap"),
  numThreads = -1
)

```

### Arguments

coefficientMatrices	list of coefficient matrices
method	method to use when calculating coefficient distances: <b>“logDiff”</b> for two coefficient matrices A and B returns $\sum(\log(1+\text{abs}(A-B)))$ <b>“wLogDiff”</b> performs the “logDiff” method with weights on the rows <b>“pa”</b> total pairs where the coefficient is present in one matrix and absent in the other (presence-absence) <b>“ap”</b> opposite comparison of pa (absence-presence)
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

### Value

distance matrix calculated from argument coefficient matrices

### Note

- the substituted y coefficient vector and the complex tip label coefficient matrix only support the “logDiff” method
- “pa” and “ap” force symmetry in the output distance matrix

**Examples**

```

library(treenomial)
library(ape)

# coefficient matrices for ten trees of 20 tips
coeffs <- treeToPoly(rmtree(10, 20), numThreads = 0)

# distance matrix from the list of coefficient matrices
d <- polyToDistMat(coeffs, method = "logDiff", numThreads = 0)

# using the absence-presence method
d <- polyToDistMat(coeffs, method = "ap", numThreads = 0)

```

---

treeDist	<i>Calculates the distance between trees</i>
----------	--

---

**Description**

Calculates the distance between two trees or a tree and a list of trees.

**Usage**

```

treeDist(
  x,
  Y,
  type = c("default", "yEvaluated", "tipLabel"),
  method = c("logDiff", "wLogDiff", "pa", "ap"),
  y,
  numThreads = -1
)

```

**Arguments**

x	single phylo object
Y	a list of phylo objects
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“tipLabel”</b> complex coefficient polynomial that utilize binary trait tip labels on the phylo objects
method	method to use when calculating coefficient distances: <b>“logDiff”</b> for two coefficient matrices A and B returns $\sum(\log(1+\text{abs}(A-B)))$ <b>“wLogDiff”</b> performs the “logDiff” method with weights on the rows



	“pa” total pairs where the coefficient is present in one matrix and absent in the other (presence-absence)
	“ap” opposite comparison of pa (absence-presence)
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

**Value**

vector of distances

**Note**

- the substituted y coefficient vector and the complex tip label coefficient matrix only support the “logDiff” method
- “pa” and “ap” force symmetry in the output distance matrix

**Examples**

```
library(treenomial)
library(ape)

# distance between one 10 tip tree and 100 trees with 30 tips

# generate the trees
tenTipTree <- rtree(10)
thirtyTipList <- rmtree(100, 30)

# find the distance
treeDist(tenTipTree, thirtyTipList, numThreads = 0)
```

---

treeJuliaSet

*Plots a Julia Set for a tree*

---

**Description**

Finds the Julia Set for the y evaluated polynomial of a tree and plots in a square image.

**Usage**

```
treeJuliaSet(
  tree,
  pixelLength = 700,
  center = 0,
  maxZ = 2,
```

```

maxIter = 100,
col = c("white", colorRampPalette(c("dodgerblue4", "lightblue"))(98), "black"),
y
)

```

### Arguments

tree	phylo object
pixellength	number of pixels on one side of the image
center	complex number giving the center of the image on the complex plane
maxZ	the max value for the real and imaginary axis
maxIter	maximum count for iterations
col	colours to be used for the image
y	the y value to evaluate the polynomial at

### Examples

```

library(treenomial)
library(ape)
treeJuliaSet(stree(5,type = "right"), y = 1+1i)

```

---

treeToDistMat

*Calculates the distance matrix from a list of phylo objects*

---

### Description

Calculates the distance matrix from a list of phylo objects

### Usage

```

treeToDistMat(
  trees,
  method = c("logDiff", "wLogDiff", "pa", "ap"),
  type = c("default", "yEvaluated", "tipLabel"),
  y,
  numThreads = -1
)

```

**Arguments**

trees	a single phylo object or a list of phylo objects
method	method to use when calculating coefficient distances: <b>“logDiff”</b> for two coefficient matrices A and B returns $\text{sum}(\log(1+\text{abs}(A-B)))$ <b>“wLogDiff”</b> performs the “logDiff” method with weights on the rows <b>“pa”</b> total pairs where the coefficient is present in one matrix and absent in the other (presence-absence) <b>“ap”</b> opposite comparison of pa (absence-presence)
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“tipLabel”</b> complex coefficient polynomial that utilize binary trait tip labels on the phylo objects
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

**Value**

a distance matrix

**Note**

- the substituted y coefficient vector and the complex tip label coefficient matrix only support the “logDiff” method
- “pa” and “ap” force symmetry in the output distance matrix

**Examples**

```
library(treenomial)
library(ape)
# distance matrix for 10 trees of 30 tips
treeToDistMat(rmtree(10, 30), method = "wLogDiff", numThreads = 0)
```

---

treeToPoly

*Convert trees to coefficient matrices*


---

### Description

Converts rooted full binary trees to tree distinguishing polynomials described with coefficient matrices.

### Usage

```
treeToPoly(
  trees,
  type = c("default", "yEvaluated", "tipLabel"),
  y,
  varLabels = FALSE,
  numThreads = -1
)
```

### Arguments

trees	a single phylo object or a list of phylo objects
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“tipLabel”</b> complex coefficient polynomial that utilize binary trait tip labels on the phylo objects
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise
varLabels	boolean for whether to add row and column names corresponding to the variables in the polynomial
numThreads	number of threads to be used, the default (-1) will use the number of cores in the machine and numThreads = 0 will only use the main thread

### Value

the resulting coefficient matrix or matrices of the form:

**“real”** a real matrix where the *i*th row, *j*th column represents the  $x^{(j-1)}y^{(i-1)}$  coefficient

**“yEvaluated”** a vector where the *k*th column represents the  $x^{(k-1)}$  coefficient

**“tipLabel”** given trees with two unique tip labels “a”, “b” a complex matrix where the *i*th row, *j*th column represents the  $a^{(i-1)}b^{(j-1)}$  coefficient

**Examples**

```

library(treenomial)
library(ape)

# generate a tree
tree <- rtree(n = 30, rooted = TRUE)

# a real coefficient matrix
treeToPoly(tree, varLabels = TRUE, numThreads = 0)

# complex coefficient vector for the tree
treeToPoly(tree, type = "yEvaluated", y = 1+1i, varLabels = TRUE, numThreads = 0)

# for a list of trees
treeToPoly(rmtree(4, 20), varLabels = TRUE, numThreads = 0)

```

---

wedge	<i>Performs the wedge operation</i>
-------	-------------------------------------

---

**Description**

Calculates the result from the wedge operation on two real coefficient matrices, two y evaluated polynomial coefficient vectors or two phylo objects.

**Usage**

```
wedge(A, B, type = c("default", "yEvaluated", "phylo"), y)
```

**Arguments**

A, B	two real coefficient matrices, complex coefficient vectors or phylo objects
type	one of: <b>“real”</b> tree distinguishing polynomials in two variables x (columns) and y (rows) <b>“yEvaluated”</b> tree distinguishing polynomials with y evaluated at a specified argument <b>“tipLabel”</b> complex coefficient polynomial that utilize binary trait tip labels on the phylo objects
y	the y value to evaluate the polynomial at when type is “yEvaluated”, ignored otherwise

**Value**

the wedge result in the same form as the arguments

**Examples**

```
library(treenomial)
library(ape)

# wedge two real coefficient matrices

leaf <- matrix(c(0,1), nrow = 1, ncol = 2)
wedge(leaf, leaf)

# wedge two complex coefficient vectors

leaf <- as.complex(c(0,1))
wedge(leaf, leaf, "yEvaluated",5)
```

# Index

`alignPoly`, [2](#)

`allTrees`, [3](#)

`plotExtremeTrees`, [4](#)

`polyDist`, [6](#)

`polyToDistMat`, [7](#)

`treeDist`, [8](#)

`treeJuliaSet`, [9](#)

`treeToDistMat`, [10](#)

`treeToPoly`, [12](#)

`wedge`, [13](#)