

# Package ‘Oncotree’

April 17, 2009

**Type** Package

**Title** Estimating oncogenetic trees

**Version** 0.3

**Date** 2009-03-10

**Author** Aniko Szabo, Lisa Pappas

**Maintainer** Aniko Szabo <aszabo@mcw.edu>

**Depends** R (>= 2.3.1), boot

**Suggests** lattice

**Description** Contains functions to construct and evaluate directed tree structures that model the process of occurrence of genetic alterations during carcinogenesis.

**License** GPL 2 or newer

**Repository** CRAN

**Date/Publication** 2009-03-12 09:28:15

## R topics documented:

Oncotree-package . . . . .	2
ancestors . . . . .	3
bootstrap . . . . .	4
distribution.oncotree . . . . .	6
error.rates<- . . . . .	7
generate.data . . . . .	8
oncotree . . . . .	9
ov.cgh . . . . .	12

<b>Index</b>	<b>13</b>
--------------	-----------

## Description

Oncogenetic trees are directed tree structures that model the process of occurrence of genetic alterations during carcinogenesis.

## Details

Package: Oncotree  
Type: Package  
Version: 1.0  
Date: 2007-01-30  
License: GPL 2 or newer

A **pure oncogenetic tree** is a directed rooted tree  $T$  with a probability  $\pi(e)$  attached to each edge  $e$  such that for every vertex there is a unique directed path from the root to it along the edges of the tree. This tree generates observations on the presence/absence of genetic events the following way: each edge  $e$  is independently retained with probability  $\pi(e)$ ; the set of vertices that are still reachable from the root gives the set of the observed genetic events.

To describe random deviations from the pure tree model an error model is added.

### Error model

1. The tumor develops according to the pure oncogenetic tree model
2. The presence/absence of each alteration is independently measured
3. If the alteration is present it is not observed with probability  $\epsilon_-$ .  
If the alteration is absent it is observed with probability  $\epsilon_+$ .

## Author(s)

Lisa Pappas, Aniko Szabo

Maintainer: Aniko Szabo <aszabo@mcw.edu>

## References

- [1] Desper R., Jiang F., Kallioniemi O.P., Moch H., Papadimitriou C.H., and Schäffer A.A. (1999) Inferring tree models for oncogenesis from comparative genome hybridization data. *Journal of Computational Biology*. **6m** 37–51. [2] Szabo, A. and Boucher, K. (2002) Estimating an oncogenetic tree when false negative and positives are present. *Mathematical Biosciences*, **176/2**, 219–236.

## Examples

```
data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)
plot(ov.tree, edge.weights="estimated")
```

---

ancestors	<i>Find ancestors within an oncogenetic tree.</i>
-----------	---

---

### Description

`ancestors` finds all the ancestors of the given vertex within the tree starting from itself up to the root. `least.common.ancestor` finds the common ancestor of two vertices that is closest to them (and farthest from the root).

### Usage

```
ancestors(otree, vertex)
least.common.ancestor(otree, v1, v2)
```

### Arguments

`otree`            An object of class `oncotree`.  
`vertex, v1, v2`    Character values giving the names of the nodes.

### Value

For `ancestors`: a character vector giving the names of the ancestors of `vertex`. The first element is `vertex`, and the last one is "Root".

For `least.common.ancestor`: a character value with the name of the least common ancestor of `v1` and `v2`.

### See Also

[oncotree.fit](#)

### Examples

```
data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)
ancestors(ov.tree, "4q-")
ancestors(ov.tree, "Xp-")
least.common.ancestor(ov.tree, "4q-", "Xp-") # "5q-"
```

**Description**

`bootstrap.oncotree` provides a set of resampling based estimates of the oncogenetic tree. Both a parametric and non-parametric approach is available. The `print` and `plot` methods provide interfaces for printing a summary and plotting the resulting set of trees.

**Usage**

```
bootstrap.oncotree(otree, R, type = c("nonparametric", "parametric"))
## S3 method for class 'boottree':
print(x, ...)
## S3 method for class 'boottree':
plot(x, minfreq=NULL, minprop=NULL, nboots=NULL, draw.orig=TRUE, draw.consensus=
      fix.nodes=FALSE, ask=(prod(par("mfrow"))<ntrees)&&
```

**Arguments**

<code>otree</code>	An object of class <code>oncotree</code> .
<code>R</code>	The number of bootstrap replicates.
<code>type</code>	The type of bootstrap - see Details for explanations.
<code>x</code>	An object of class <code>boottree</code> - the output of <code>bootstrap.oncotree</code>
<code>minfreq</code>	A lower limit on the occurrence frequency of the tree in “boottree” for plotting. By default, all unique trees are plotted, which can lead to a large number of plots.
<code>minprop</code>	A lower limit on the occurrence proportion of the tree in “boottree” for plotting.
<code>nboots</code>	A lower limit on the number of bootstrapped trees plotted.
<code>draw.orig</code>	logical; if TRUE the original tree is plotted.
<code>draw.consensus</code>	logical; if TRUE the consensus tree is plotted (see Details).
<code>fix.nodes</code>	logical; if TRUE, the nodes for all trees are kept in the same position. If <code>node.coords</code> is passed as an argument to <code>plot.oncotree</code> , then those coordinates are used for all trees, otherwise the coordinates computed for the original tree are used.
<code>ask</code>	logical; if TRUE, the user is <i>asked</i> before each plot, see <code>par(ask=)</code> .
<code>...</code>	Ignored for <code>print</code> . Passed to <code>plot.oncotree</code> for the <code>plot</code> method.

## Details

**Parametric bootstrap:** This approach assumes that the model is correct. Based on `otree`, a random data set is generated `R` times using `generate.data`. An oncogenetic tree is fitted to each of these random data sets.

**Non-parametric bootstrap:** The samples (rows) from the data associated with the tree are re-sampled with replacement `R` times, each time obtaining a data set with the same sample size. An oncogenetic tree is fitted to each of these resampled data sets.

For both approaches, a *consensus tree* that assigns to each vertex the parent that occurs most frequently in the bootstrapped trees, is also computed.

## Value

For `bootstrap.oncotree`: an object of class `boottree` with the following components:

<code>original</code>	The parent component of the original tree ( <code>otree</code> ).
<code>consensus</code>	A numeric vector with the <code>parent\$parent.num</code> component of the consensus tree - this defines the tree structure uniquely.
<code>parent.freq</code>	A matrix giving the number of trees with each possible child-parent edge. The rows correspond to children while the column to parents.
<code>tree.list</code>	A data frame with each row representing a unique tree obtained during the bootstrap. The 'Tree' variable contains the <code>parent\$parent.num</code> component of the tree (each pasted into one dot-separated string), while the 'Freq' variable gives the frequency of the tree among the <code>R</code> bootstrap replicates.
<code>type</code>	A character value with the type of the bootstrap performed.

For `print.boottree`: the original object is returned invisibly. It prints a summary showing the number of replicates, the number of unique trees found, and the number of times that the original tree was obtained.

For `plot.oncotree`: nothing is returned. It is used for its side effect of producing a sequence of plots of the bootstrapped trees. Specifically, it plots the original tree (if `draw.orig=TRUE`), the consensus tree (if `draw.consensus=TRUE`), and then the other trees by frequency of occurrence. To limit the number of bootstrapped trees plotted, specify exactly one of `minfreq`, `minprop` or `nboots`. By default, if the session is interactive, the user is asked for confirmation before each new tree is drawn. To avoid this, either use `ask=FALSE` in the function call, or set up a layout that fits all the trees.

## Author(s)

Lisa Pappas, Aniko Szabo

## See Also

[oncotree.fit](#)

**Examples**

```

data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)
set.seed(43636)
ov.b1 <- bootstrap.oncotree(ov.tree, R=100, type="parametric")
ov.b1
opar <- par(mfrow=c(3,2), mar=c(2,0,0,0))
plot(ov.b1, nboots=4)
plot(ov.b1, nboots=4, fix.nodes=TRUE)
par(opar)

```

---

distribution.oncotree

*Find the event distribution defined by an oncogenetic tree*

---

**Description**

distribution.oncotree calculates the joint distribution of the events defined by the tree, while marginal.distr calculates the marginal probability of occurrence of each event.

**Usage**

```

distribution.oncotree(otree, with.probs = TRUE, with.errors=FALSE,
  edge.weights=if (with.errors) "estimated" else "observed")
marginal.distr(otree, with.errors = TRUE,
  edge.weights=if (with.errors) "estimated" else "observed")

```

**Arguments**

otree	An object of class oncotree.
with.probs	A logical value specifying if only the set of possible outcomes should be returned (if TRUE), or the associated probabilities of occurrence as well.
with.errors	A logical value specifying whether false positive and negative error rates should be incorporated into the distribution.
edge.weights	A choice of whether the observed or estimated edge transition probabilities should be used in the calculation of probabilities. See <a href="#">oncotree.fit</a> for explanation of the difference. By default, estimated edge transition probabilities if with.errors=TRUE and the observed ones if with.errors=FALSE.

**Value**

For distribution.oncotree: a data frame each row of which gives a possible outcome.

For marginal.distr: a named numeric vector - the names are the event names (+ 'Root') and the values are the corresponding marginal probability of occurrence.

**Author(s)**

Aniko Szabo

**See Also**[oncotree.fit](#)**Examples**

```

data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)

#joint distribution
jj <- distribution.oncotree(ov.tree, edge.weights="obs")
head(jj)
jj.eps <- distribution.oncotree(ov.tree, with.errors=TRUE)
head(jj.eps)

#marginal distribution
marginal.distr(ov.tree, with.error=FALSE)
#marginal distribution calculated from the joint
apply(jj[1:ov.tree$nmult], 2, function(x){sum(x*jj$Prob)})

##Same with errors incorporated
#marginal distribution
marginal.distr(ov.tree, with.error=TRUE)
#marginal distribution calculated from the joint
apply(jj.eps[1:ov.tree$nmult], 2, function(x){sum(x*jj.eps$Prob)})

```

---

```
error.rates<-
```

*Set the error rates of an oncotree manually*

---

**Description**

Allows to set the false positive and false negative error rate associated with an object of class `oncotree` to values other than those found by the optimization in [oncotree.fit](#). The estimated edge transition probabilities are updated appropriately.

**Usage**

```
error.rates(x) <- value
```

**Arguments**

`x` An object of class `oncotree`.

`value` A numeric vector of length 2. The false positive error rate will be set to `value[1]`, while the false negative error rate to `value[2]`.

**See Also**[oncotree.fit](#)**Examples**

```

data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)
ov.tree
error.rates(ov.tree) <- c(0,0)
ov.tree

```

---

generate.data

*Generate random data from an oncogenetic tree*


---

**Description**

Generates random event occurrence data based on an oncogenetic tree model.

**Usage**

```

generate.data(N, otree, with.errors=TRUE,
             edge.weights=if (with.errors) "estimated" else "observed")

```

**Arguments**

N	The required sample size.
otree	An object of the class <code>oncotree</code> .
with.errors	A logical value specifying whether false positive and negative errors should be applied.
edge.weights	A choice of whether the observed or estimated edge transition probabilities should be used in the calculation of probabilities. See <a href="#">oncotree.fit</a> for explanation of the difference. By default, estimated edge transition probabilities if <code>with.errors=TRUE</code> and the observed ones if <code>with.errors=FALSE</code> .

**Details**

Technically, the distribution generated by the tree is calculated exactly (using [distribution.oncotree](#)), and the observations are generated by sampling this distribution. Thus if `N` is small and `with.errors=TRUE`, it might be faster to avoid the computational overhead of calculating the entire distribution, but rather generate data not including false positive/negatives and then randomly ‘corrupt’ it (see Examples below).

**Value**

A data set where each row is an independent observation.

**Author(s)**

Aniko Szabo

**See Also**[oncotree.fit](#)**Examples**

```

data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh)

set.seed(7365)
rd <- generate.data(200, ov.tree, with.errors=TRUE)

#corrupt data - useful for small N
system.time({
  rd2 <- generate.data(20, ov.tree, with.errors=FALSE);
  epos <- ov.tree$eps[["epos"]];
  eneg <- ov.tree$eps[["eneg"]];
  corrupt.data <- matrix(rbinom(prod(dim(rd2)), size=1, p=ifelse(rd2==0, epos, 1-eneg)),
                        nr=nrow(rd2), nc=ncol(rd2),
                        dimnames=list(NULL, names(rd2)))
})
system.time(generate.data(20, ov.tree, with.errors=TRUE))

```

oncotree

*Build and display an oncogenetic tree***Description**

Build a directed tree structure to model the process of occurrence of genetic alterations (events) in carcinogenesis. The model is described in more detail in [Oncotree-package](#). Methods for printing a short summary, displaying the tree on an R plot, and producing latex code for drawing the tree (using the ‘pstricks’ and ‘pst-tree’ LaTeX packages) are provided.

**Usage**

```

oncotree.fit(dataset, error.fun = function(x, y){sum((x - y)^2)})
## S3 method for class 'oncotree':
print(x, ...)
## S3 method for class 'oncotree':
plot(x, edge.weights = c("none", "observed", "estimated"),
      edge.digits=2, node.coords=NULL, plot=TRUE, cex = par("cex"),
                                             col.edge=par("col"), col.text=par("col"),
pmtree.oncotree(x, edge.weights=c("none", "observed", "estimated"), edge.digits=2,
                shape=c("none", "oval", "circle", "triangle", "diamond"),
                pmtree.options=list(arrows=">", treefit="loose", arrowsscale="1.5

```

**Arguments**

<code>dataset</code>	A data frame or a matrix with variable names as a listing of genetic events taking on binary values indicating missing (0) or present (1). Each row is an independent sample.
<code>error.fun</code>	A function of two variables that measures the deviation of the observed marginal frequencies of the events (which will be the first argument in the call) from the estimated ones. The false positive and negative error rates are obtained by minimizing <code>error.fun</code> . If <code>error.fun=NULL</code> is used, the error rates are not estimated.
<code>x</code>	An object of class <code>oncotree</code> .
<code>edge.weights</code>	Choice of edge weights to show on the plot.
<code>edge.digits</code>	The number of significant digits to use when displaying edge weights.
<code>node.coords</code>	A matrix with node-coordinates or <code>NULL</code> if the coordinates should be computed automatically (default).
<code>plot</code>	Logical; indicates whether the tree should be plotted.
<code>cex</code>	Scaling factor for the text in the nodes.
<code>col.edge</code>	color of the tree edges.
<code>col.text</code>	color of the node label.
<code>col.weight</code>	color of the edge weights.
<code>...</code>	Ignored for <code>print</code> . For <code>plot</code> these can be graphical parameters passed to <code>lines</code> when the edges are drawn
<code>shape</code>	The shape of the node in the <code>pst-tree</code> representation.
<code>pstree.options</code>	Additional options for <code>pst-tree</code> . See the <code>pstricks</code> documentation for possible values.

**Details**

'pst-tree' is a very flexible package, and very detailed formatting of the tree is possible. `pstree.oncotree` provides some default settings for drawing trees, but they can be easily overridden: most options can be set in `pstree.options`, while the appearance of the tree nodes can be controlled by defining a one-parameter `\lab` command that gives the desired appearance. For example, if red, non-mathematical test is desired in an oval, you could use `\newcommand{\lab}[1]{\Toval[name=#1]{\red #1}}`.

**Value**

For `oncotree.fit`: an object of class `oncotree` which has components

<code>data</code>	data frame used, after dropping events with zero observed frequency, and adding a column for the artificial 'Root' node
<code>nmut</code>	number of tree nodes: the number genetic events present in <code>data</code> +1 for the 'Root' node
<code>parent</code>	a list containing information about the tree structure with the following components

`child` a character vector of the event names starting with 'Root'  
`parent` a character vector of the names of the parents of `child`  
`parent.num` a numeric vector with column indices corresponding to `parent`  
`obs.weights` raw edge transition probabilities  $P(\text{child}|\text{parent})$   
`est.weights` edge transition probabilities adjusted for the error rates `eps`  
`level` a numeric vector of the depth of each node in the tree (1 for the root, 2 for its children, etc.)  
`numchild` a numeric vector giving the number of children for each node  
`levelnodes` a numeric vector of the number of nodes found at each level of the tree  
`levelgrp` a character matrix with its rows giving the ordered nodes at each level  
`eps` a numeric vector of length two showing the estimated false positive and negative error rates (if `error.fun` is not NULL). Do not modify directly, but rather through `error.rates<-`.

For `print.oncotree`:

the original object is returned invisibly. It prints a summary showing the number of nodes, the parent-child relationships, and the false positive and negative error rates.

For `plot.oncotree`:

a matrix with node-coordinates is returned invisibly. The column names of the matrix are the names of the nodes/events (including 'Root'), the rows gives the x- and y-coordinates, respectively. This matrix provides a valid input for `node.coords`. If `plot=TRUE`, a plot of the tree is produced.

For `pstree.oncotree`:

a character string with the LaTeX code needed to draw a tree. `\usepackage{pstricks,pst-tree}` is required in the preamble of the LaTeX file, and it should be processed through a PostScript intermediary (DVIPS or similar) and not through PDFLaTeX.

### Author(s)

Lisa Pappas

### References

Szabo, A. and Boucher, K. (2002) Estimating an oncogenetic tree when false negative and positives are present. *Mathematical Biosciences*, 176/2, 219-236.

### See Also

[bootstrap.oncotree](#), [error.rates<-](#), [generate.data.ancestors](#), [distribution.oncotree](#)

### Examples

```

data(ov.cgh)
ov.tree <- oncotree.fit(ov.cgh, error.fun=function(x,y){max(abs(x-y))})
ov.tree
nodes <- plot(ov.tree, edge.weights="est")
#move the Root node to the left

```

```

nodes["x", "Root"] <- nodes["x", "8q+"]
plot(ov.tree, node.coords=nodes)
#output for pstricks+pst-tree
pstree.oncotree(ov.tree, edge.weights="obs", shape="oval")

```

---

ov.cgh

*Ovarian cancer CGH data*


---

### Description

This is a data set obtained using the comparative genomic hybridization technique (CGH) on samples from papillary serous cystadenocarcinoma of the ovary. Only the seven most commonly occurring events are given.

### Usage

```
data(ov.cgh)
```

### Format

A data frame with 87 observations on the following 7 variables.

**8q+** a 0/1 indicator of the presence of the ‘8q+’ event

**3q+** a 0/1 indicator of the presence of the ‘3q+’ event

**5q-** a 0/1 indicator of the presence of the ‘5q-’ event

**4q-** a 0/1 indicator of the presence of the ‘4q-’ event

**8p-** a 0/1 indicator of the presence of the ‘8p-’ event

**1q+** a 0/1 indicator of the presence of the ‘1q+’ event

**Xp-** a 0/1 indicator of the presence of the ‘Xp-’ event

### Details

The CGH technique uses fluorescent staining to detect abnormal (increased or decreased) number of DNA copies. Often the results are reported as a gain or loss on a certain arm, without further distinction for specific regions. It is common to denote a change in DNA copy number on a specific chromosome arm by prefixing a “-” sign for decrease and a “+” for increase. Thus, say, -3q denotes abnormally low DNA copy number on the q arm of the 3rd chromosome.

### Source

<http://www.ncbi.nlm.nih.gov/sky/>

### Examples

```

data(ov.cgh)
heatmap(data.matrix(ov.cgh), Colv=NA, scale="none", col=c("gray90", "red"))

```

# Index

- \*Topic **datagen**
    - generate.data, 7
  - \*Topic **datasets**
    - ov.cgh, 11
  - \*Topic **distribution**
    - distribution.ancotree, 5
  - \*Topic **graphs**
    - ancotree, 9
  - \*Topic **hplot**
    - ancotree, 9
  - \*Topic **models**
    - ancestors, 2
    - bootstrap, 3
    - distribution.ancotree, 5
    - error.rates<-, 7
    - generate.data, 7
    - ancotree, 9
  - \*Topic **nonparametric**
    - bootstrap, 3
  - \*Topic **package**
    - Oncotree-package, 1
  - \*Topic **tree**
    - ancotree, 9
- ancestors, 2, 11
- bootstrap, 3
- bootstrap.ancotree, 11
- distribution.ancotree, 5, 8, 11
- error.rates<-, 7, 10, 11
- generate.data, 4, 7, 11
- least.common.ancestor  
(ancestors), 2
- marginal.distr  
(distribution.ancotree), 5
- Oncotree (*Oncotree-package*), 1
- ancotree, 9
- Oncotree-package, 9
- Oncotree-package, 1
- ancotree.fit, 3, 5–8
- ov.cgh, 11
- par, 4
- plot.boottree (*bootstrap*), 3
- plot.ancotree, 4
- plot.ancotree (*ancotree*), 9
- print.boottree (*bootstrap*), 3
- print.ancotree (*ancotree*), 9
- pstree.ancotree (*ancotree*), 9