

Package ‘delt’

June 17, 2009

Version 0.8.0

Date 2009-06-15

Title Estimation of multivariate densities with adaptive histograms

Author Jussi Klemela <klemela@oulu.fi>

Maintainer Jussi Klemela <klemela@oulu.fi>

Depends denpro (>= 0.9)

Description The package implements methods for estimating multivariate densities: adaptive histograms (greedy histograms and CART-histograms), stagewise minimization, and bootstrap aggregation are included in the package.

License GPL (>= 2)

URL <http://www.rni.helsinki.fi/~jsk/delt>

Repository CRAN

Date/Publication 2009-06-17 12:05:07

R topics documented:

delt-package	2
densplit	4
eval.bagg	5
eval.cart	7
eval.greedy	8
eval.pick	9
eval.stage	10
eval.stage.gauss	11
intpcf	12
lefrig2par	13
lstseq.bagg	14
lstseq.cart	16
lstseq.greedy	17

makebina	18
partition	19
plotparti	20
prune	21
scaspa	22
supp	23

Index	25
--------------	-----------

delt-package	<i>Estimation of multivariate densities with adaptive histograms</i>
--------------	--

Description

The package implements methods for estimating multivariate densities: adaptive histograms (greedy histograms and CART-histograms), stagewise minimization, and bootstrap aggregation are provided.

Details

```

Package:  delt
Version:  0.8.0
Date:     2009-06-17
Depends:  R
License:  GPL version 2 or newer
URL:      http://www.rni.helsinki.fi/~jsk/delt
Packaged: Mon Jun 15 16:34:17 2009; jsk
Built:    R 2.6.2; i486-pc-linux-gnu; 2009-06-15 16:34:35; unix

```

Greedy histograms: eval.greedy, lstseq.greedy.

CART-histograms: eval.cart, lstseq.cart.

CART histograms step by step: densplit, prune, eval.pick.

Bootstrap aggregation of histograms: eval.bagg, lstseq.bagg.

Stagewise minimization: eval.stage, eval.stage.gauss.

Other utilities: partition, plotparti.

Tree transformation: lefrig2par, makebina.

Miscellaneous: intpcf, supp.

Index:

densplit	Calculation of an overfitting histogram
eval.bagg	Returns a bootstrap aggregation of adaptive histograms
eval.cart	Calculates a CART histogram
eval.greedy	Returns a greedy histogram

eval.pick	Returns a subtree of an evaluation tree
eval.stage	Returns a stagewise minimization estimate
eval.stage.gauss	Returns a 1D Gaussian mixture density estimate
intpcf	Calculates the integral of a piecewise constant function
lefrig2par	Transforms an evaluation tree so that it can be plotted with the "plottree" function of package "denpro"
lstseq.bagg	Calculates a scale of bootstrap aggregated histograms
lstseq.cart	Calculates a scale of CART histograms
lstseq.greedy	Calculates a scale of greedy histograms
makebina	Transforms and evaluation tree to the tree object of R
partition	Finds the partition generated by an evaluation tree
plotparti	Draws a partition
prune	Prepares for pruning an overfitting evaluation tree
scaspa	Finds the number of modes of histograms which are obtained by pruning an overfitting histogram
supp	Returns the bounding box of observations

Author(s)

Jussi Klemela <klemela@oulu.fi>

Maintainer: Jussi Klemela <klemela@oulu.fi>

Examples

```
# Generate the data
dendat<-sim.data(n=500,seed=5,type="mulmodII")

# Calculate the estimates
eva<-eval.greedy(dendat,leaf=16)
eva<-eval.cart(dendat,leaf=16)
eva<-eval.bagg(dendat,B=3,leaf=12,prune="on")
eva<-eval.stage(dendat,leaf=10,M=3)

# Draw the estimates
lst<-leafsfirst(eva)
plotvolu(lst)
```

```
dp<-draw.pcf(eva, pnum=c(60, 60))
persp(dp$x, dp$y, dp$z, theta=-20, phi=30)
```

densplit

Calculation of an overfitting histogram

Description

The function returns an overfitting histogram when a data matrix is given as an input. The output is an evaluation tree which is grown with greedy growing. The evaluation tree defines a partition of the sample space. The evaluation tree may be pruned to get a density estimate.

Usage

```
densplit(dendat, minobs=NULL, leaf=0, method="loglik",
         splitscan=0, seedf=1, suppo=NULL)
```

Arguments

dendat	n*d data matrix
minobs	non-negative integer; splitting of a bin will be continued if the bin contains "minobs" or more observations
leaf	internal (maximal number of leafs in the evaluation tree)
method	"loglik" or "projec"; the contrast function
splitscan	internal (random selection of splits)
seedf	internal
suppo	2*d vector of real numbers; the rectangle to be splitted; the rectangle has to contain the data

Value

Returns an evaluation tree as a list of vectors.

direc	integer in 1,...,d; variable which is splitted
split	real number; splitting point
mean	nonnegative number; value of the histogram on the rectangle corresponding to the node
nelem	nonnegative integer; number of observations in the rectangle corresponding to the node
ssr	real number; value of the likelihood criterion
volume	non-negative number; volume of the rectangle corresponding to the node
left	non-negative integer; link to the left child, 0 if terminal node

right	non-negative integer; link to the right child, 0 if terminal node
low	the lower vertice of the rectangles
upp	the upper vertice of the rectangles
N	the number of grid points at each direction
support	the support of the histogram

Author(s)

Jussi Klemela

See Also[prune](#), [eval.pick](#)**Examples**

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

treeseq<-prune(et)
treeseq$leafs
len<-length(treeseq$leafs)

leaf<-treeseq$leafs[len-10]
leaf
etsub<-eval.pick(treeseq,leaf=leaf)

dp<-draw.pcf(etsub)
persp(dp$x,dp$y,dp$z,phi=25,theta=-120)
```

`eval.bagg`*Returns a bootstrap aggregation of adaptive histograms*

Description

Returns a bootstrap aggregation of CART-histograms or greedy histograms.

Usage

```
eval.bagg(dendat, B, leaf, minobs = NULL, seed = 1, sample = "bagg",
prune = "off", splitscan = 0, seedf = 1, scatter = 0, src = "c",
method = "loglik")
```

Arguments

dendat	n*d data matrix
B	positive integer; the number of aggregated histograms
leaf	the cardinality of the partitions of the aggregated histograms
minobs	non-negative integer; a property of aggregated histograms; splitting of a bin will be continued if the bin contains "minobs" or more observations
seed	the seed for the random number generation of the random selection of the bootstrap sample
sample	"bagg" or "worpl"; the bootstrapping method; "worpl" for the n/2-out-of-n without replacement; "bagg" for n-out-of-n with replacement
prune	"on" or "off"; if "on", then CART-histograms will be aggregated; if "off", then greedy histograms will be aggregated
splitscan	internal (how many splits will be used for random split selection)
seedf	internal (seed for random split selection)
scatter	internal (random perturbation of observations)
src	internal ("c" or "R" code)
method	"loglik" or "projec"; the empirical risk is either the log-likelihood or the L2 empirical risk

Value

An evaluation tree

Author(s)

Jussi Klemela

See Also

[lstseq.bagg](#), [eval.cart](#), [eval.greedy](#)

Examples

```
dendat<-sim.data(n=600,seed=5,type="mulmodII")

leaf<-7      # number of leaves in the histograms
seed<-1      # seed for choosing bootstrap samples
sample="worpl" # without-replacement bootstrap
prune="on"   # we use CART-histograms
B<-50       # the number of histograms in the average

eva<-eval.bagg(dendat,B,leaf,seed=seed,sample=sample,prune=prune)

dp<-draw.pcf(eva,pnum=c(60,60))
persp(dp$x,dp$y,dp$z,theta=-20,phi=30)
```

`eval.cart`*Calculates a CART histogram*

Description

Calculates a CART histogram. The estimate is represented as an evaluation tree. An CART histogram is a multivariate adaptive histogram which is obtained by pruning an evaluation tree of an overfitting histogram.

Usage

```
eval.cart(dendat, leaf, minobs = NULL)
```

Arguments

<code>dendat</code>	n*d data matrix
<code>leaf</code>	positive integer; the cardinality of the partition of the histogram
<code>minobs</code>	non-negative integer; splitting of a bin of the overfitting histogram will be continued if the bin contains "minobs" or more observations

Details

The partition of the histogram may not contain exactly "leaf" rectangles: the cardinality of the partition is as close as possible to "leaf"

Value

An evaluation tree

Author(s)

Jussi Klemela

See Also

[lstseq.cart](#), [densplit](#)

Examples

```
dendat<-sim.data(n=600,seed=5,type="mulmodII")
eva<-eval.cart(dendat,16)

dp<-draw.pcf(eva,pnum=c(60,60))
persp(dp$x,dp$y,dp$z,theta=-20,phi=30)
```

eval.greedy	Returns a greedy histogram
-------------	----------------------------

Description

Returns a greedy histogram. A greedy histogram is grown stagewise by minimizing an empirical risk functional.

Usage

```
eval.greedy(dendat, leaf, method = "loglik", minobs = NULL, bound = 0,
suppo = NULL)
```

Arguments

dendat	n*d data matrix
leaf	the (maximal) number of rectangles in the partition of the histogram
method	"loglik" or "projec"; the empirical risk is either the log-likelihood or the L2 empirical risk
minobs	non-negative integer; splitting of a bin will be continued if the bin contains "minobs" or more observations
bound	internal
suppo	2*d vector of real numbers; the rectangle to be splitted; the rectangle has to contain the data

Value

An evaluation tree

Author(s)

Jussi Klemela

See Also

[lstseq.greedy](#), [partition](#)

Examples

```
dendat<-sim.data(n=400,seed=5,type="mulmodII")
eva<-eval.greedy(dendat,leaf=15)

dp<-draw.pcf(eva,pnum=c(60,60))
persp(dp$x,dp$y,dp$z,theta=-20,phi=30)
```

`eval.pick`*Returns a subtree of an evaluation tree*

Description

Returns a subtree of an evaluation tree. The subtree has a specified number of leafs. The evaluation tree is calculated by "densplit" function. To find out the possible values for the number of leaves we use function "prune".

Usage

```
eval.pick(treeseq, leaf)
```

Arguments

<code>treeseq</code>	an overfitting evaluation tree with information on the possible pruning nodes; output of function "prune"
<code>leaf</code>	positive integer; number of leaves in the subtree to be returned

Value

Returns an evaluation tree, see the documentation of function "eval.cart"

Author(s)

Jussi Klemela

See Also

[densplit](#), [prune](#)

Examples

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

treeseq<-prune(et)
treeseq$leafs
len<-length(treeseq$leafs)

leaf<-treeseq$leafs[len-10]
leaf
etsub<-eval.pick(treeseq,leaf=leaf)

dp<-draw.pcf(etsub)
persp(dp$x,dp$y,dp$z,phi=25,theta=-120)
```

`eval.stage` *Returns a stagewise minimization estimate*

Description

Returns a stagewise minimization estimate. A stagewise minimization estimator is a convex combination of greedy histograms. The convex combination is constructed by a stagewise minimization of an empirical risk functional.

Usage

```
eval.stage(dendat, leaf, M, pis = NULL, mcn = dim(dendat)[1],
minobs = NULL, seedi = 1, method = "projec", bound = 0)
```

Arguments

<code>dendat</code>	n*d data matrix
<code>leaf</code>	the (maximal) number of rectangles in the partition of the greedy histograms
<code>M</code>	the number of histograms in the convex combination
<code>pis</code>	the vector of weights of the convex combination
<code>mcn</code>	the size of the Monte Carlo sample used in the numerical integration in calculating the empirical risk functional
<code>minobs</code>	non-negative integer; splitting of a bin of a greedy histogram will be continued if the bin contains "minobs" or more observations
<code>seedi</code>	the seed for the generation of the Monte Carlo sample
<code>method</code>	"loglik" or "projec"; the empirical risk is either the log-likelihood or the L2 empirical risk
<code>bound</code>	internal

Value

An evaluation tree

Author(s)

Jussi Klemela

See Also

[eval.greedy](#), [eval.stage.gauss](#)

Examples

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
leaf<-13 # the number of leafs of the greedy histograms
M<-5     # the number of greedy histograms

pcf<-eval.stage(dendat,leaf=leaf,M=M)

dp<-draw.pcf(pcf,pnum=c(120,120))
persp(dp$x,dp$y,dp$z,ticktype="detailed",phi=25,theta=-120)
```

```
eval.stage.gauss Returns a 1D Gaussian mixture density estimate
```

Description

Estimates a 1D density with a mixture of Gaussians. The mixture is found by minimizing the L2 empirical risk in a stagewise manner.

Usage

```
eval.stage.gauss(dendat, M, mugrid, siggrid = 1, sigeka = TRUE, src = "c",
  sampstart=FALSE, boost=FALSE, N=60)
```

Arguments

dendat	n-vector of 1D observations
M	integer ≥ 1 ; the number of mixture components in the estimate
mugrid	a vector of real numbers; the range for the means of mixture components
siggrid	a vector of real numbers; the range of possible standard deviations in the mixture components
sigeka	TRUE or FALSE; if TRUE, then the standard deviation of the first mixture component is equal to 1, otherwise the standard deviation of the first mixture component is found by minimization
src	"R" or "c"; if "R", then the R-code is used, otherwise the c-code is used
sampstart	internal
boost	internal
N	postive integer; the number of evaluation points

Value

A piecewise constant function with the additional components:

muut	vector of real numbers; the means of the mixture components
sigit	vector of positive real numbers; the standard deviations of the mixture components
curmix	a probability vector; the weights of the mixture components

Author(s)

Jussi Klemela

References

Jussi Klemela (2005). Density Estimation with Stagewise Optimization of the Empirical Risk

See Also

[eval.stage](#)

Examples

```
dendat<-sim.data(n=100,type="1d2modal",seed=1)

mugrid<-seq(-1,5,0.3) # grid of mu-values
siggrid<-seq(0.2,2,0.2) # grid of sigma-values
M<-17 # number of mixture components
pcf<-eval.stage.gauss(dendat,M,mugrid,siggrid)

dp<-draw.pcf(pcf)
plot(dp$x,dp$y,type="l")

# draw the estimate with the help of package "denpro"
#N<-100
#pcf2<-pcf.func("mixt",N,sig=pcf$sigit,M=pcf$muut,p=pcf$curmix)
#pnum<-100
#dm<-draw.pcf(pcf2,pnum=pnum)
#plot(dm$x,dm$y,type="l")
```

intpcf

Calculates the integral of a piecewise constant function

Description

Calculates the integral of a piecewise constant function.

Usage

```
intpcf(pcf)
```

Arguments

pcf piecewise constant function

Value

Real number; the integral of the piecewise constant function

Author(s)

Jussi Klemela

Examples

```
dendat<-sim.data(n=100,seed=5,type="mulmodII")
eva<-eval.greedy(dendat,leaf=5)
intpcf(eva)
```

lefrig2par

Transforms an evaluation tree so that it can be plotted with the "plot-tree" function of package "denpro"

Description

Evaluation trees are trees which are implemented with "left" and "right" pointers. We transform this tree representation to the representation with "parent" pointers (level set tree), so that it can be plotted with the "plottree" function of package "denpro". For example, functions "densplit" and "eval.pick" return evaluation trees.

Usage

```
lefrig2par(et)
```

Arguments

et evaluation tree; result of "densplit", "eval.pick", "eval.cart", ...; see the documentation of "eval.cart"

Value

Returns a level set tree:

parent	parent links
level	height of the node
center	determines the ordering of the nodes
volume	determines the horizontal positioning of the nodes

Author(s)

Jussi Klemela

See Also

[densplit](#), [eval.pick](#)

Examples

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

l1st<-lefrig2par(et)

plottree(l1st)
```

l1stseq.bagg

Calculates a scale of bootstrap aggregated histograms

Description

Calculates a scale of bootstrap aggregated histograms. The estimates in the sequence are calculated with function "eval.bagg".

Usage

```
l1stseq.bagg(dendat, B, l1stree=NULL, level = NULL,
maxleaf = NULL, leafseq = NULL,
minobs = NULL, seed = 1, sample = "bagg", prune = "off",
splitscan = 0, seedf = 1, scatter = 0, src = "c", method = "loglik")
```

Arguments

dendat	n*d data matrix
B	positive integer; the number of aggregated histograms
maxleaf	the maximal cardinality of the partitions of the histograms in the sequence
l1stree	if NULL, then level set trees are not calculated
level	if NULL, then shape trees are not calculated; if positive number, then it is the level of the level sets for which the shape trees are calculated
leafseq	a vector giving the cardinalities of the partitions of the aggregated histograms
minobs	non-negative integer; a property of aggregated histograms; splitting of a bin will be continued if the bin contains "minobs" or more observations
seed	the seed for the random number generation of the random selection of the bootstrap sample
sample	"bagg" or "worpl"; the bootstrapping method; "worpl" for the n/2-out-of-n without replacement; "bagg" for n-out-of-n with replacement
prune	"on" or "off"; if "on", then CART-histograms will be aggregated; if "off", then greedy histograms will be aggregated
splitscan	internal (how many splits will be used for random split selection)
seedf	internal (seed for random split selection)

scatter	internal (random perturbation of observations)
src	internal ("c" or "R" code)
method	"loglik" or "projec"; the empirical risk is either the log-likelihood or the L2 empirical risk

Value

A list with components

l1stseq	a list of level set trees
pcfseq	a list of piecewise constant functions
stseq	a list of shape trees
hseq	a vector of smoothing parameters corresponding to the estimates in the sequence; the smoothing parameter is the cardinality of the partitions of the aggregated histograms

Author(s)

Jussi Klemela

See Also

[eval.bagg](#)

Examples

```
dendat<-sim.data(n=200,seed=1,type="mulmodII")

seed<-1      # seed for choosing bootstrap samples
sample="worpl" # without-replacement bootstrap
prune="on"   # we use CART-histograms
B<-2        # the number of histograms in the average

estiseq<-l1stseq.bagg(dendat,B,maxleaf=10,lstree=TRUE,
                      seed=seed,sample=sample,prune=prune)

mt<-modegraph(estiseq)

plotmodet(mt)

#scaletable(estiseq)
```

`lstseq.cart`*Calculates a scale of CART histograms*

Description

Calculates a scale of CART histograms. The histograms in the scale have partitions of growing cardinality. Returns a sequence of estimates as piecewise constant functions and optionally as level set trees. Optionally a shape tree for a level set of each estimate is calculated.

Usage

```
lstseq.cart(treeseq, maxleaf=NULL, lstree=NULL, level = NULL,  
indvec = NULL)
```

Arguments

<code>treeseq</code>	output of function "prune"
<code>maxleaf</code>	the maximal cardinality of the partitions of the histograms in the sequence
<code>lstree</code>	if NULL, then level set trees are not calculated
<code>level</code>	if NULL, then shape trees are not calculated; if positive number, then it is the level of the level sets for which the shape trees are calculated
<code>indvec</code>	a vector of indices; chooses a subset of the complete sequence of subhistograms of the overfitting histogram

Value

A list with components

<code>lstseq</code>	a list of level set trees
<code>pcfseq</code>	a list of piecewise constant functions
<code>stseq</code>	a list of shape trees
<code>hseq</code>	a vector of smoothing parameters corresponding to the members in the sequences; the smoothing parameter is the cardinality of the partition

Author(s)

Jussi Klemela

See Also

[densplit](#), [prune](#)

Examples

```
dendat<-sim.data(n=200,seed=1,type="mulmodII")
et<-densplit(dendat)
treeseq<-prune(et)

estiseq<-l1stseq.cart(treeseq,maxleaf=20,lstree=TRUE)

mt<-modegraph(estiseq)

plotmodet(mt)

#scaletable(estiseq)
```

l1stseq.greedy	<i>Calculates a scale of greedy histograms</i>
----------------	--

Description

Calculates a scale of greedy histograms. The histograms in the scale have a partition of growing cardinality. Returns a sequence of estimates as piecewise constant functions and optionally as level set trees. Optionally a shape tree for a level set of each estimate is calculated.

Usage

```
l1stseq.greedy(dendat, maxleaf, lstree = NULL, level = NULL)
```

Arguments

dendat	n*d data matrix
maxleaf	integer>1; the scale consists of histograms whose partitions have cardinality 1,...,maxleaf
lstree	if NULL, then level set trees are not calculated
level	if NULL, then shape trees are not calculated; if positive number, then it is the level of the level sets for which the shape trees are calculated

Value

A list with components

l1stseq	a list of level set trees
pcfseq	a list of piecewise constant functions
stseq	a list of shape trees
hseq	a vector of smoothing parameters corresponding to the members in the sequences; the smoothing parameter is the cardinality of the partition

Author(s)

Jussi Klemela

See Also

[eval.greedy](#),

Examples

```
dendat<-sim.data(n=200,seed=1,type="mulmodII")
estiseq<-lstseq.greedy(dendat,maxleaf=20,lstree=TRUE)

mt<-modegraph(estiseq)

plotmodet(mt)

#scaletable(estiseq)
```

makebina

Transforms and evaluation tree to the tree object of R

Description

Evaluation trees are such trees that are implemented with "left" and "right" pointers. We transform this tree representation to the tree object of the package "tree", so that it can be plotted by "plot.tree" function from package "tree" or by "draw.tree" function from package "maptree".

Usage

```
makebina(et)
```

Arguments

et evaluation tree; result of "densplit", "eval.pick", "eval.cart", ...; see the documentation of "eval.cart"

Value

Returns an object of class tree.

Author(s)

Jussi Klemela

See Also

[densplit](#), [eval.pick](#)

Examples

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)
mb<-makebina(et)

set.seed(1)
dendat<-matrix(rnorm(20),10)
et<-densplit(dendat,minobs=2)
mb<-makebina(et)

#library(tree)
#plot.tree(mb)

#library(maptree)
#draw.tree(mb)
```

partition

Finds the partition generated by an evaluation tree

Description

Finds the partition generated by an evaluation tree. An evaluation tree makes a recursive partition of a rectangle. Functions "eval.cart", "densplit",... return evaluation trees.

Usage

```
partition(et, grid=TRUE, zerorecs=FALSE)
```

Arguments

et	an evaluation tree; output of "eval.cart", "densplit", ...
grid	TRUE or FALSE; whether the true coordinates or relative coordinates are used
zerorecs	TRUE or FALSE; whether the rectangles where the density vanishes are included; (evaluation trees are used by package "delt" to represent density functions)

Value

Returns a list with the following elements.

values	vector whose length is equal to the number of rectangles in the partition; value of the function on the corresponding rectangle
recs	recnum*(2*d) matrix; recnum is the number of rectangles in the partition and d is the dimension of the observations. The rows of "recs" describe the rectangles. Column (2*j-1) gives the lower value for the j:th interval and (2*j):th column gives upper value for the j:th interval, j=1,...,d.
support	the rectangle which is partitioned

Author(s)

Jussi Klemela

See Also[plotparti](#),**Examples**

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

pa<-partition(et)

plotparti(pa)
```

<code>plotparti</code>	<i>Draws a partition</i>
------------------------	--------------------------

Description

Draws the partition calculated with function "partition".

Usage

```
plotparti(pa, d1 = NULL, d2 = NULL, dendat = NULL, restri = NULL,
pch = 21, support = pa$support, col = "black", cex.axis = 1)
```

Arguments

<code>pa</code>	partition; output of function "partition"
<code>d1</code>	integer 1,...,d; for the case the partition is a partition of a higher than 2 dimensional rectangle, "d1" is the first direction of the partition
<code>d2</code>	integer 1,...,d; for the case the partition is a partition of a higher than 2 dimensional rectangle, "d2" is the second direction of the partition
<code>dendat</code>	n*d data matrix; if given as an argument it will also be plotted
<code>restri</code>	internal
<code>pch</code>	symbol for plotting "dendat"; see function "points"
<code>support</code>	the bounds of the partition
<code>col</code>	color of the lines of the partition
<code>cex.axis</code>	magnification factor for the axis annotation; see "par"

Value

A plot at the graphics window

Author(s)

Jussi Klemela

See Also[partition](#)**Examples**

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

pa<-partition(et)

plotparti(pa)
```

prune

*Prepares for pruning an overfitting evaluation tree***Description**

Finds a sequence of nodes of an overfitting evaluation tree which are candidates to be the pruning nodes. Pruning a tree means removing a branch starting from a node.

Usage

```
prune(et)
```

Arguments

`et` an evaluation tree; output of "eval.cart", "densplit", ...

Value

A list containing the following components.

<code>tree</code>	the original tree which was given as the input
<code>delnodes</code>	vector giving a sequence of nodes in the order in which we should prune the branches starting from these nodes
<code>delend</code>	vector whose length is the number of subtrees of the original tree. With the help of "delend" we define the subtrees. Elements of "delend" define a sequence of nodes from "delnodes" in the following way: (1:delend[1]) is the first sequence, (delend[1]+1:delend[2]) is the second sequence, and so on. Then, <i>i</i> :th subtree is the result of pruning branches away whose roots are the nodes which are the first delend[<i>i</i>] elements of delnodes.
<code>leafs</code>	vector whose length is the number of subtrees of the original tree; number of leafs of the subtrees

`alfa` vector whose length is the number of subtrees of the original tree; value of the corresponding `alfa` (complexity parameter) for every subtree

`loglik` vector whose length is the number of subtrees of the original tree; the value of the likelihood criterion for the subtree

Author(s)

Jussi Klemela

See Also

[densplit](#), [eval.pick](#)

Examples

```
dendat<-sim.data(n=200,seed=5,type="mulmodII")
et<-densplit(dendat)

treeseq<-prune(et)
treeseq$leafs
len<-length(treeseq$leafs)

leaf<-treeseq$leafs[len-10]
leaf
etsub<-eval.pick(treeseq,leaf=leaf)

dp<-draw.pcf(etsub)
persp(dp$x,dp$y,dp$z,phi=25,theta=-120)
```

`scaspa`

Finds the number of modes of histograms which are obtained by pruning an overfitting histogram

Description

Function "densplit" returns an overfitting histogram as an evaluation tree. Function "prune" finds the candidate nodes for pruning. Function "scaspa" finds the number of modes (local maxima) in the histograms which are obtained by using these candidate nodes for pruning.

Usage

```
scaspa(treeseq, bind, eind)
```

Arguments

treeseq	a list returned by function "prune"
bind	integer in 1:subnum, where subnum is the number of elements in field "leafs" of "treeseq". We need bind<eind.
eind	integer in 1:subnum, where subnum is the number of elements in field "leafs" of "treeseq". We need bind<eind.

Value

List with the following vectors whose length is (eind-bind+1)

moodlkm	number of local maxima for each subtree
alfas	value of the smoothing parameter alpha for each subtree
leafnums	number of leaves for each subtree

Author(s)

Jussi Klemela

See Also

[densplit](#), [prune](#)

Examples

```
set.seed(1)
dendat<-matrix(rnorm(20),10)
minlkm<-2
et<-densplit(dendat,minlkm)
treeseq<-prune(et)
treeseq$leafs

scaspa(treeseq,1,5)
```

supp

Returns the bounding box of observations

Description

Returns the smallest rectangle containing the observations. The sides of the rectangle are parallel to the coordinate axis.

Usage

```
supp(dendat, epsi=0, blown=FALSE)
```

Arguments

dendat n*d data matrix
epsi positive number: option to return the smallest rectangle such that the epsi-shrinkage of the rectangle contains the observations
blown internal

Value

2*d vector: (2*i-1)-element, $i=1, \dots, d$, is the start of the i:th interval and (2*i)-element is the end of the i:th interval

Author(s)

Jussi Klemela

Examples

```
set.seed(1)
dendat<-matrix(rnorm(20),10)
supp(dendat)
```

Index

*Topic **hplot**

lefrig2par, 13
makebina, 18
partition, 19
plotparti, 20

*Topic **math**

intpcf, 12

*Topic **multivariate**

densplit, 3
eval.bagg, 5
eval.cart, 6
eval.greedy, 7
eval.pick, 8
eval.stage, 9
lstseq.bagg, 14
lstseq.cart, 16
lstseq.greedy, 17
prune, 21
supp, 23

*Topic **smooth**

delt-package, 1
densplit, 3
eval.bagg, 5
eval.cart, 6
eval.greedy, 7
eval.pick, 8
eval.stage, 9
eval.stage.gauss, 11
lstseq.bagg, 14
lstseq.cart, 16
lstseq.greedy, 17
prune, 21
scaspa, 22

delt (*delt-package*), 1
delt-package, 1
densplit, 3, 7, 9, 13, 16, 18, 22, 23

eval.bagg, 5, 15
eval.cart, 6, 6

eval.greedy, 6, 7, 10, 18
eval.pick, 4, 8, 13, 18, 22
eval.stage, 9, 12
eval.stage.gauss, 10, 11

intpcf, 12

lefrig2par, 13
lstseq.bagg, 6, 14
lstseq.cart, 7, 16
lstseq.greedy, 8, 17

makebina, 18

partition, 8, 19, 21
plotparti, 20, 20
prune, 4, 9, 16, 21, 23

scaspa, 22
supp, 23