

# Package ‘scaleboot’

February 15, 2012

**Type** Package

**Title** Approximately Unbiased P-values via Multiscale Bootstrap

**Version** 0.3-3

**Date** 2010-05-14

**Author** Hidetoshi Shimodaira

**Maintainer** Hidetoshi Shimodaira <shimo@is.titech.ac.jp>

**Depends** R (>= 2.0.0)

**Suggests** pvclust (>= 1.2.0), snow

**Description** Calculating approximately unbiased (AU) p-values from multiscale bootstrap probabilities.

**License** GPL (>= 2)

**URL** <http://www.is.titech.ac.jp/~shimo/prog/scaleboot/>

**Repository** CRAN

**Date/Publication** 2010-05-16 16:21:31

## R topics documented:

scaleboot-package . . . . .	2
coef . . . . .	2
interface . . . . .	3
lung73 . . . . .	5
mam15 . . . . .	8
plot.scaleboot . . . . .	15
relltest . . . . .	17
sbaic . . . . .	19
sbconf . . . . .	20
sbit . . . . .	24

sboptions . . . . .	27
sbpsi . . . . .	28
sbpval . . . . .	30
scaleboot . . . . .	31
summary.scaleboot . . . . .	34

<b>Index</b>	<b>37</b>
--------------	-----------

---

scaleboot-package	<i>Approximately Unbiased P-values via Multiscale Bootstrap</i>
-------------------	-----------------------------------------------------------------

---

### Description

Calculating approximately unbiased (AU) p-values from multiscale bootstrap probabilities.

### Details

For a complete list, use `library(help="scaleboot")`.

The methodology is described in Shimodaira (2008). For the use of `scaleboot`, Shimodaira (2008) may be referenced.

Further information is available in the following vignette:

`usesb` Multiscale Bootstrap using Scaleboot Package

### Author(s)

Hidetoshi Shimodaira

Maintainer: Hidetoshi Shimodaira <shimo@is.titech.ac.jp>

I thank Paul A. Sheridan for his comments to improve the manual pages.

### References

Shimodaira, H. (2008). Testing Regions with Nonsmooth Boundaries via Multiscale Bootstrap, *Journal of Statistical Planning and Inference*, 138, 1227-1241. (<http://dx.doi.org/10.1016/j.jspi.2007.04.001>).

---

coef	<i>Extract Model Coefficients</i>
------	-----------------------------------

---

### Description

Extract the estimated parameters from "scaleboot" or "scalebootv" objects.

**Usage**

```
## S3 method for class 'scaleboot'  
coef(object, sd=FALSE, ...)  
  
## S3 method for class 'scalebootv'  
coef(object, ...)
```

**Arguments**

<code>object</code>	an object used to select a method.
<code>...</code>	further arguments passed to or from other methods.
<code>sd</code>	logical. Should standard errors be returned as well?

**Value**

The `coef` method for the class "scaleboot" returns a matrix consisting of row vectors of beta's for models. If `sd=TRUE`, it returns a list with components `estimate` and `sd` for the beta matrix and its standard error respectively.

**Author(s)**

Hidetoshi Shimodaira

**See Also**

[sbfit](#)

**Examples**

```
data(mam15)  
a <- mam15.relltest[["t4"]] # an object of class "scaleboot"  
coef(a) # print the estimated beta values  
coef(a, sd=TRUE) # with sd
```

---

interface

*Interface to External Packages*

---

**Description**

Interface for other packages such as CONSEL (phylogenetic inference), and pvclust (hierarchical clustering)

**Usage**

```

read.mt(file, tlab="t")

read.ass(file, identity=TRUE, tlab="t", elab="e")

read.cnt(file)

## S3 method for class 'pvclust'
sbfit(x, ...)

sbpvclust(x, mbs, k=3, select="average", ...)

```

**Arguments**

<code>file</code>	character of a file name to be read.
<code>identity</code>	logical. Should the identity association be included?
<code>tlab</code>	character for basename of tree labels.
<code>elab</code>	character for basename of edge labels.
<code>x</code>	an object of class "pvclust".
<code>mbs</code>	an object of class "scalebootv".
<code>k</code>	numeric of $k$ for a AU p-value.
<code>select</code>	character of model name (such as "poly.3") or one of "average" and "best".
<code>...</code>	further arguments passed to or from other methods.

**Details**

CONSEL is a program package consisting of small programs written in the C language for assessing the confidence of phylogenetic tree selection. Some functions for interfacing with CONSEL are: `read.mt`, `read.ass`, and `read.cnt` for reading, respectively, `mt`, `ass`, and `cnt` format. Once `mt` file is read, we can calculate improved versions of approximately unbiased p-values by `relltestin` `scaleboot` instead of CONSEL.

`pvclust` is a R package for hierarchical clustering with p-values. Functions for interface to `pvclust` are: `sbfit` method for an object of class "pvclust" to convert it to "scalebootv" class, and `sbpvclust` for writing back the result to a "pvclust" object.

**Value**

`read.mt` returns a matrix of dimension sequence-length by tree-number. If `identity=FALSE`, then `read.ass` returns a list containing components `x` for edge→tree associations and `y` for tree→edge associations. If `identity=TRUE`, `read.ass` returns a list vector of edge→tree associations, where the identity associations for tree→tree are included. `read.cnt` returns a list containing components `bps`, `nb`, and `sa` to be used for `sbfit`. The list also contains components `cnt`, `id`, and `val`.

`sbfit.pvclust` returns an object of class "scalebootv". `sbpvclust` returns an object of class "pvclust".

**Author(s)**

Hidetoshi Shimodaira

**References**

Shimodaira, H. and Hasegawa, M. (2001). CONSEL: for assessing the confidence of phylogenetic tree selection, *Bioinformatics*, 17, 1246-1247 (software is available from <http://www.is.titech.ac.jp/~shimo/prog/consel/>).

Suzuki, R. and Shimodaira, H. (2006). pvclust: An R package for hierarchical clustering with p-values, *Bioinformatics*, 22, 1540-1542 (software is available from CRAN or <http://www.is.titech.ac.jp/~shimo/prog/pvclust/>).

**See Also**

[relltest](#)

**Examples**

```
## Not run:
## reading CONSEL files
## sample files are found in mam15 subdirectory
## see help(mam15) for details
mam15.mt <- read.mt("mam15.mt")
mam15.ass <- read.mt("mam15.ass")
mam15.cnt <- read.mt("mam15.cnt")

## End(Not run)

## Not run:
## see help(lung73) for details
data(lung73)
lung73.sb <- sbfit(lung73.pvclust,cluster=cl) # model fitting
lung73.new <- sbpvclust(lung73.pvclust,lung73.sb) # au <- k.3

## End(Not run)
```

---

lung73

*Clustering of 73 Lung Tumors*

---

**Description**

Bootstrapping hierarchical clustering of the DNA microarray data set of 73 lung tissue samples each containing 916 observed genes.

**Usage**

```
data(lung73)

lung73.pvclust

lung73.sb
```

**Format**

lung73.pvclust is an object of class "pvclust" defined in **pvclust** of Suzuki and Shimodaira (2006).

lung73.sb is an object of class "scalebootv" of length 72.

**Details**

The microarray dataset of Garber et al. (2001) is reanalyzed in Suzuki and Shimodaira (2006), and is found in data(lung) of the **pvclust** package. We reanalyze it, again, by the script shown in Examples. The result of pvclust is stored in lung73.pvclust, and model fitting to bootstrap probabilities by the **scaleboot** package is stored in lung73.sb. The AU p-values obtained by using the **scaleboot** package are sometimes very different from those obtained by the **pvclust** package. For example, pvclust with default parameter value gave AU p-value of 0.70 for Edge-67, but the sbfit gives AU p-value (named "k.3") of 0.95 for the same edge. Note that the raw bootstrap probability (i.e., the ordinary bootstrap probability with scale=1) is 0.04.

The AU p-values for all nodes are shown by the summary method,

```
> summary(lung73.sb[60:70])
```

Corrected P-values (percent):

	raw	k.1	k.2	k.3	model	aic
60	20.21 (0.40)	20.29 (0.18)	71.40 (0.20)	78.98 (0.44)	sing.3	80.46
61	58.45 (0.49)	55.08 (0.17)	63.15 (0.24)	56.34 (0.38)	poly.3	575.85
62	95.68 (0.20)	95.92 (0.10)	98.64 (0.10)	98.61 (0.12)	poly.3	-12.01
63	58.31 (0.49)	57.30 (0.17)	82.09 (0.20)	81.74 (0.28)	poly.3	20.74
64	15.81 (0.36)	15.58 (0.16)	75.36 (0.21)	84.86 (0.37)	sing.3	71.47
65	2.96 (0.17)	2.80 (0.07)	76.73 (0.51)	94.88 (0.20)	sing.3	33.34
66	15.75 (0.36)	15.92 (0.16)	78.02 (0.20)	87.98 (0.29)	sing.3	7.30
67	3.63 (0.19)	3.31 (0.07)	77.02 (0.47)	95.10 (0.17)	sing.3	25.11
68	26.20 (0.44)	27.06 (0.17)	83.06 (0.18)	84.90 (0.27)	poly.3	8.67
69	29.49 (0.46)	29.65 (0.17)	75.37 (0.22)	75.83 (0.34)	poly.3	-14.09
70	28.31 (0.45)	29.04 (0.19)	76.62 (0.17)	81.54 (0.37)	sing.3	0.99

Shown above are four types of p-values as well as selected model and AIC values. "raw" is the ordinary bootstrap probability, "k.1" is equivalent to "raw" but calculated from the multiscale bootstrap, "k.2" is equivalent to the third-order AU p-value of CONSEL, and finally "k.3" is an improved version of AU p-value. By default, we use "k.3" when copying back the p-values to an object of class "pvclust".

See Examples below for details.

## Note

The microarray dataset is not included in `data(lung73)`, but it is found in `data(lung)` of the **pvclust** package.

## Source

Garber, M. E. et al. (2001) Diversity of gene expression in adenocarcinoma of the lung, *Proceedings of the National Academy of Sciences*, 98, 13784-13789 (dataset is available from [http://genome-www.stanford.edu/lung\\_cancer/aden/](http://genome-www.stanford.edu/lung_cancer/aden/)).

## References

Suzuki, R. and Shimodaira, H. (2006). `pvclust`: An R package for hierarchical clustering with p-values, *Bioinformatics*, 22, 1540-1542 (software is available from CRAN or <http://www.is.titech.ac.jp/~shimo/prog/pvclust/>).

## See Also

[sbpvclust](#), [sbfit.pvclust](#)

## Examples

```
## Not run:
## script to create lung73.pvclust and lung73.sb
## multiscale bootstrap resampling of hierarchical clustering
library(pvclust)
data(lung)
sa <- 9^seq(-1,1,length=13) # wider range of scales than pvclust default
lung73.pvclust <- pvclust(lung,r=1/sa,nboot=10000)
lung73.sb <- sbfit(lung73.pvclust) # model fitting

## End(Not run)

## Not run:
## Parallel version of the above script
## parPvclust took 80 mins using 40 cpu's
library(snow)
library(pvclust)
data(lung)
cl <- makeCluster(40) # launch 40 cpu's
sa <- 9^seq(-1,1,length=13) # wider range of scales than pvclust default
lung73.pvclust <- parPvclust(cl,lung,r=1/sa,nboot=10000)
lung73.sb <- sbfit(lung73.pvclust,cluster=cl) # model fitting

## End(Not run)

## replace au/bp entries in pvclust object
data(lung73)
lung73.new <- sbpvclust(lung73.pvclust,lung73.sb) # au <- k.3

## Not run:
```

```

library(pvclust)
plot(lung73.new) # draw dendrogram with the new au/bp values
pvrect(lung73.new)

## End(Not run)

## diagnose edges 61,...,69
lung73.sb[61:69] # print fitting details
plot(lung73.sb[61:69]) # plot curve fitting
summary(lung73.sb[61:69]) # print au p-values
## diagnose edge 67
lung73.sb[[67]] # print fitting
plot(lung73.sb[[67]],legend="topleft") # plot curve fitting
summary(lung73.sb[[67]]) # print au p-values

```

---

mam15

*Mammal Phylogenetic Analysis for 15 Trees*


---

## Description

Phylogenetic analysis of six mammal species for 15 trees.

## Usage

```

data(mam15)

mam15.mt

mam15.ass

mam15.relltest

```

## Format

mam15.mt is a matrix of size 3414 \* 15. The (i,j) element is the site-wise log-likelihood value at site-i for tree-j for i=1,...,3414, and j=1,...,15.

mam15.ass is a list of length 25 for association vectors. The components are t1, t2, ..., t15 for trees, and e1, e2, ..., e10 for edges.

mam15.relltest is an object of class "relltest" of length 25.

## Details

An example of phylogenetic analysis of six mammal species: *Homo sapiens* (human), *Phoca vitulina* (harbor seal), *Bos taurus* (cow), *Oryctolagus cuniculus* (rabbit), *Mus musculus* (mouse), *Didelphis virginiana* (opossum). The data is stored in the file 'mam15.aa', which contains amino acid sequences of length N=3414 for the six species obtained from mtDNA (see Note below). We consider 15 tree topologies of the six mammals as stored in the file 'mam15.tp1';

```

((Homsa,(Phovi,Bosta)),Orycu,(Musmu,Didvi)); t1
(Homsa,Orycu,((Phovi,Bosta),(Musmu,Didvi))); t2
(Homsa,((Phovi,Bosta),Orycu),(Musmu,Didvi)); t3
(Homsa,(Orycu,Musmu),((Phovi,Bosta),Didvi)); t4
((Homsa,(Phovi,Bosta)),(Orycu,Musmu),Didvi); t5
(Homsa,((Phovi,Bosta),(Orycu,Musmu)),Didvi); t6
(Homsa,(((Phovi,Bosta),Orycu),Musmu),Didvi); t7
(((Homsa,(Phovi,Bosta)),Musmu),Orycu,Didvi); t8
(((Homsa,Musmu),(Phovi,Bosta)),Orycu,Didvi); t9
(Homsa,Orycu,(((Phovi,Bosta),Musmu),Didvi)); t10
(Homsa,(((Phovi,Bosta),Musmu),Orycu),Didvi); t11
((Homsa,((Phovi,Bosta),Musmu)),Orycu,Didvi); t12
(Homsa,Orycu,(((Phovi,Bosta),Didvi),Musmu)); t13
((Homsa,Musmu),Orycu,((Phovi,Bosta),Didvi)); t14
((Homsa,Musmu),((Phovi,Bosta),Orycu),Didvi); t15

```

The log-likelihood values are calculated using the PAML software (Ziheng 1997) for phylogenetic inference. The two files 'mam15.aa' and 'mam15.tpl' are fed into PAML to generate the file 'mam15.lnf' of site-wise log-likelihood values.

Using the CONSEL software (Shimodaira and Hasegawa 2001), we convert 'mam15.lnf' and 'mam15.tpl' to a format suitable for the **scaleboot** package. We do not use CONSEL for calculating AU p-values, but use it only for file conversion. We type

```

seqmt --paml mam15.lnf
treeass --outgroup 6 mam15.tpl > mam15.log

```

The first line above generates 'mam15.mt', which is a simple text file containing a matrix of site-wise log-likelihood values. The second line above generates 'mam15.ass' and 'mam15.log', which contain information regarding which edges are included in a tree. A part of 'mam15.log' is as follows.

```

# leaves: 6
6
 1 Homsa
 2 Phovi
 3 Bosta
 4 Orycu
 5 Musmu
 6 Didvi

# base edges: 10
10 6

 123456
1 +++--- ;
2 ++++-- ;
3 +--+-- ;
4 -+++-- ;

```

```

5 ---+--+ ;
6 +---+--+ ;
7 -++++--+ ;
8 ++++--+ ;
9 +----+--+ ;
10 -++++--+ ;

```

The above defines edges named e1,...e10 (base edges) as clusters for six mammal species. For example, e1 = +++— = (Homsa, Phovi, Bosta).

The converted files are read by the **scaleboot** package in R:

```

mam15.mt <- read.mt("mam15.mt")
mam15.ass <- read.ass("mam15.ass")

```

mam15.mt is a matrix of size 3414 \* 6 for the site-wise log-likelihood values. For testing trees, we need only mam15.mt. mam15.ass is used for testing edges, and it is a list of length 25 for association vectors for t1,t2,...,t15, and e1,e2,...,e10. For example, mam15.ass\$t1 = 1, indicating tree "t1" is included in tree "t1", and mam15.ass\$e1 = c(1, 5, 8), indicating edge "e1" is included in trees "t1", "t5", and "t8".

Multiscale bootstrap resampling is performed by the function `relltest`. The simplest way to get AU p-values for trees is:

```

mam15.trees <- relltest(mam15.mt) # resampling and fitting
summary(mam15.trees) # calculates AU p-values

```

The `relltest` returns an object of class "relltest". It calls the function `scaleboot` internally with the number of bootstrap replicates `nb=10000`, and takes about 20 mins. Typically, `nb=10000` is large enough, but it would be safe to use larger value, say `nb=100000` as in the examples below.

Note that the default value of scales in `relltest` has a much wider range than that of `CONSEL`. It is `sa=9^seq(-1,1,length=13)` for `relltest`, and is `sa=1/seq(from=0.5, to=1.4, by=0.1)` for `CONSEL`.

The `mam15.relltest` object in `data(mam15)` is similar to `mam15.trees` above, but is also calculated for edges using `mam15.ass`. We can extract the result for trees by

```

mam15.trees <- mam15.relltest[1:15]

```

The results for trees stored in the `mam15.trees` object above are in the order specified in the columns of `mam15.mt`. To sort it by increasing order of the log-likelihood difference, we can type

```

stat <- attr(mam15.trees,"stat") # the log-likelihood differences
o <- order(stat) # sort it in increasing order
mam15.trees <- mam15.trees[o] # same as mam15.trees in Examples

```

Results of the fitting are shown by using the print method.

```
> mam15.trees
```

```
Test Statistic, and Shimodaira-Hasegawa test:
```

	stat	shtest	
t1	-2.66	94.51	(0.07)
t3	2.66	80.25	(0.13)
t2	7.40	57.85	(0.16)
t5	17.57	17.30	(0.12)
t6	18.93	14.32	(0.11)
t7	20.11	11.49	(0.10)
t4	20.60	10.98	(0.10)
t15	22.22	7.34	(0.08)
t8	25.38	3.31	(0.06)
t14	26.32	3.29	(0.06)
t13	28.86	1.71	(0.04)
t9	31.64	0.61	(0.02)
t11	31.75	0.57	(0.02)
t10	34.74	0.20	(0.01)
t12	36.25	0.12	(0.01)

```
Multiscale Bootstrap Probabilities (percent):
```

	1	2	3	4	5	6	7	8	9	10	11	12	13
t1	86	81	77	73	68	63	58	52	46	41	36	31	28
t3	14	19	23	27	30	32	32	31	30	27	25	22	20
t2	0	0	0	0	1	2	4	5	7	9	10	11	11
t5	0	0	0	0	0	1	1	2	3	5	6	6	7
t6	0	0	0	0	1	2	3	5	6	7	8	9	9
t7	0	0	0	0	0	0	0	1	2	3	4	5	5
t4	0	0	0	0	0	1	2	3	4	4	5	6	6
t15	0	0	0	0	0	0	0	0	1	1	2	2	3
t8	0	0	0	0	0	0	0	0	0	0	1	1	1
t14	0	0	0	0	0	0	0	1	1	2	3	4	4
t13	0	0	0	0	0	0	0	0	0	0	1	1	2
t9	0	0	0	0	0	0	0	0	0	0	0	1	1
t11	0	0	0	0	0	0	0	0	0	0	0	1	1
t10	0	0	0	0	0	0	0	0	0	0	0	0	0
t12	0	0	0	0	0	0	0	0	0	0	0	0	0

```
Numbers of Bootstrap Replicates:
```

1	2	3	4	5	6	7	8	9	10	11	12	13
1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05	1e+05

```
Scales (Sigma Squared):
```

1	2	3	4	5	6	7	8	9	10	11	12	13
0.1111	0.1603	0.2311	0.3333	0.4808	0.6933	1	1.442	2.080	3	4.327	6.241	9.008

```
AIC values of Model Fitting:
```

poly.1	poly.2	poly.3	sing.3

```

t1 89483.40 964.33 964.75 966.33
t3 75434.97 1750.22 1306.50 1752.22
t2 29361.29 403.41 36.33 -6.21
t5 23893.19 260.44 -0.22 -14.11
t6 35791.26 330.50 4.31 -2.49
t7 15221.10 93.59 -10.33 -12.04
t4 29790.60 453.95 5.22 -7.57
t15 6874.98 46.16 -10.48 -17.08
t8 1747.13 -6.88 -12.39 -13.68
t14 10905.94 131.48 2.65 -10.79
t13 3411.26 27.66 -8.30 -15.14
t9 1494.58 19.46 -13.78 -15.86
t11 914.42 -19.65 -19.71 -19.61
t10 259.68 -14.79 -17.27 -16.76
t12 178.79 -19.19 -19.61 -19.30

```

The AU p-values are shown by the summary method.

```
> summary(mam15.trees)
```

Corrected P-values (percent):

	raw	k.1	k.2	k.3	model	aic
t1	57.58 (0.16)	56.16 (0.04)	74.55 (0.05)	74.55 (0.05)	poly.2	964.33
t3	31.86 (0.15)	30.26 (0.05)	46.41 (0.09)	45.33 (0.13)	poly.3	1306.50
t2	3.68 (0.06)	3.68 (0.03)	12.97 (0.20)	16.12 (0.45)	sing.3	-6.21
t5	1.34 (0.04)	1.33 (0.02)	7.92 (0.25)	10.56 (0.56)	sing.3	-14.11
t6	3.18 (0.06)	3.15 (0.02)	13.15 (0.21)	15.86 (0.44)	sing.3	-2.49
t7	0.49 (0.02)	0.52 (0.01)	3.66 (0.21)	4.75 (0.42)	sing.3	-12.04
t4	1.55 (0.04)	1.53 (0.02)	10.54 (0.27)	14.84 (0.66)	sing.3	-7.57
t15	0.08 (0.01)	0.07 (0.00)	1.11 (0.19)	1.85 (0.48)	sing.3	-17.08
t8	0.00 (0.00)	0.00 (0.00)	0.04 (0.03)	0.07 (0.07)	sing.3	-13.68
t14	0.22 (0.01)	0.23 (0.01)	2.76 (0.26)	4.59 (0.71)	sing.3	-10.79
t13	0.02 (0.00)	0.01 (0.00)	0.50 (0.20)	1.30 (0.83)	sing.3	-15.14
t9	0.00 (0.00)	0.00 (0.00)	0.23 (0.05)	1.41 (0.29)	sing.3	-15.86
t11	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	poly.3	-19.71
t10	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	poly.3	-17.27
t12	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	poly.3	-19.61

The p-values for 15 trees are shown above. "raw" is the ordinary bootstrap probability, "k.1" is equivalent to "raw" but calculated from the multiscale bootstrap, "k.2" is equivalent to the third-order AU p-value of CONSEL, and finally "k.3" is an improved version of AU p-value.

The details for each tree are shown by extracting the corresponding element. For example, details for the seventh largest tree in the log-likelihood value ("t4") is obtained by

```
> mam15.trees[[7]] # same as mam15.trees$t4
```

Multiscale Bootstrap Probabilities (percent):

```

1  2  3  4  5  6  7  8  9  10  11  12  13
0.00 0.00 0.01 0.08 0.27 0.80 1.55 2.55 3.58 4.42 5.22 6.00 6.38

```

```

Numbers of Bootstrap Replicates:
 1     2     3     4     5     6     7     8     9    10    11    12    13
1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05 1e+05

Scales (Sigma Squared):
 1     2     3     4     5     6     7 8     9    10 11    12    13
0.1111 0.1603 0.2311 0.3333 0.4808 0.6933 1 1.442 2.080 3 4.327 6.241 9.008

Coefficients:
          beta0          beta1          beta2
poly.1 2.8388 (0.0048)
poly.2 1.8556 (0.0061) 0.3259 (0.0019)
poly.3 1.7157 (0.0085) 0.4508 (0.0061) -0.0152 (0.0007)
sing.3 1.6178 (0.0153) 0.5435 (0.0143) 0.3261 (0.0201)

Model Fitting:
          rss      df pfit   aic
poly.1 29814.60 12 0.0000 29790.60
poly.2  475.95 11 0.0000  453.95
poly.3   25.22 10 0.0050    5.22
sing.3   12.43 10 0.2571   -7.57

Best Model:  sing.3
> summary(mam15.trees[[7]])

Raw Bootstrap Probability:  1.55 (0.04)

Corrected P-values (percent):
          k.1          k.2          k.3          aic
poly.1 0.23 (0.00) 0.23 (0.00) 0.23 (0.00) 29790.60
poly.2 1.46 (0.02) 6.30 (0.09) 6.30 (0.09)  453.95
poly.3 1.57 (0.02) 9.50 (0.21) 10.57 (0.27)    5.22
sing.3 1.53 (0.02) 10.54 (0.27) 14.84 (0.66)   -7.57

Best Model:  sing.3

> plot(mam15.trees[[7]],legend="topleft")

```

The plot diagnostics found in the bottom line are especially useful for confirming which model is fitting best.

See other examples below.

## Note

Dataset files for phylogenetic inference are found at <http://www.is.titech.ac.jp/~shimo/prog/scaleboot/>. For Unix users, download 'mam15-files.tgz', and for Windows users download 'mam15-files.zip'. This dataset was originally used in Shimodaira and Hasegawa (1999).

## Source

H. Shimodaira and M. Hasegawa (1999). Multiple comparisons of log-likelihoods with applications to phylogenetic inference, *Molecular Biology and Evolution*, 16, 1114-1116.

## References

Yang, Z. (1997). PAML: a program package for phylogenetic analysis by maximum likelihood, *Computer Applications in BioSciences*, 13:555-556 (software is available from <http://abacus.gene.ucl.ac.uk/software/paml.html>).

Shimodaira, H. and Hasegawa, M. (2001). CONSEL: for assessing the confidence of phylogenetic tree selection, *Bioinformatics*, 17, 1246-1247 (software is available from <http://www.is.titech.ac.jp/~shimo/prog/consel/>).

## See Also

[relltest](#), [summary.scalebootv](#), [read.mt](#), [read.ass](#).

## Examples

```
data(mam15)

## show the results for trees and edges
mam15.relltest # print stat, shtest, bootstrap probabilities, and AIC
summary(mam15.relltest) # print AU p-values

## Not run:
## simpler script to create mam15.trees
mam15.mt <- read.mt("mam15.mt")
mam15.ass <- read.ass("mam15.ass")
mam15.trees <- relltest(mam15.mt,nb=100000)

## End(Not run)

## Not run:
## script to create mam15.relltest
mam15.mt <- read.mt("mam15.mt")
mam15.ass <- read.ass("mam15.ass")
mam15.relltest <- relltest(mam15.mt,nb=100000,ass=mam15.ass)

## End(Not run)

## Not run:
## Parallel version of the above script (but different in random seed)
## It took 13 mins (40 cpu's of Athlon MP 2000+)
mam15.mt <- read.mt("mam15.mt")
mam15.ass <- read.ass("mam15.ass")
library(snow)
cl <- makeCluster(40)
mam15.relltest <- relltest(mam15.mt,nb=100000,ass=mam15.ass,cluster=cl)

## End(Not run)
```

---

plot.scaleboot	<i>Plot Diagnostics for Multiscale Bootstrap</i>
----------------	--------------------------------------------------

---

## Description

plot method for class "scaleboot".

## Usage

```
## S3 method for class 'scaleboot'
plot(x, models=NULL, select=NULL, sort.by=c("aic","none"),
     k=NULL, s=NULL, sp=NULL, lambda=NULL,
     xval = c("square", "inverse","sigma"),
     yval = c("psi", "zvalue", "pvalue"), xlab = NULL,
     ylab = NULL, log.xy = "", xlim = NULL, ylim = NULL,
     add = F, length.x = 300, main=NULL,
     col =1:6, lty = 1:5, lwd = par("lwd"), ex.pch=2:7,
     pch = 1, cex = 1, pt.col = col[1], pt.lwd = lwd[1],
     legend.x = NULL, inset = 0.1, ...)

## S3 method for class 'summary.scaleboot'
plot(x, select="average",
     k=x$parex$k, s=x$parex$s, sp=x$parex$sp, lambda=x$parex$lambda, ...)

## S3 method for class 'scalebootv'
plot(x, models=attr(x, "models"), sort.by="none", ...)

## S3 method for class 'summary.scalebootv'
plot(x, select="average", ...)

## S3 method for class 'scaleboot'
lines(x, z, models=names(x$fi), k=NULL, s=NULL, sp=NULL, lambda=NULL,
      length.x=z$length.x, col=z$col, lty=z$lty, lwd=z$lwd, ... )

sblegend(x="topright", y=NULL, z, inset=0.1, ...)
```

## Arguments

x an object used to select a method. For sblegend, x is a numeric or character such as "lefttop" or "righttop", which is passed to legend.

models	character vector of model names. Numeric is also allowed.
select	"average", "best", or one of the fitted models.
sort.by	"aic" or "none".
k	k for extrapolation.
s	s for extrapolation.
sp	sp for extrapolation.
lambda	a numeric of specifying the type of p-values; Bayesian (lambda=0) Frequentist (lambda=1).
xval	specifies x-axis. "square" for $\sigma^2$ , "inverse" for $1/\sigma$ , "sigma" for $\sigma$ .
yval	specifies y-axis. "zvalue" for $\psi(\sigma^2 \beta)/\sigma$ or $qnorm(1-bp[i])$ , "pvalue" for $1 - \Phi(\psi(\sigma^2 \beta)/\sigma)$ or $bp[i]$ , "psi" for $\psi(\sigma^2 \beta)$ or $\sqrt{sa[i]}*qnorm(1-bp[i])$ .
xlab	label for x-axis.
ylab	label for y-axis.
log.xy	character to specify log-scale. "", "x", "y", or "xy".
xlim	range for x-axis.
ylim	range for y-axis.
add	logical for adding another plot.
length.x	the number of segments to draw curves.
main	for title.
col	color for model curves.
lty	lty for model curves.
lwd	lwd for model curves.
ex.pch	pch for extrapolation.
pch	pch for bp points.
cex	cex for bp points.
pt.col	col for bp points.
pt.lwd	lwd for bp points.
legend.x	passed to sblegend as the first argument.
...	further arguments passed to or from other methods.
z	output from previous plot.scaleboot.
y	numeric passed to legend.
inset	inset distance from the margins, which is passed to legend.

### Details

The plot method plots bootstrap probabilities and calls the lines method, which draws fitted curves for models.

### Author(s)

Hidetoshi Shimodaira

**See Also**

[sbfitt](#).

**Examples**

```
data(mam15)
## a single plot
a <- mam15.relltest[["t4"]] # an object of class "scaleboot"
plot(a,legend="topleft") # x=sigma^2, y=psi
plot(a,xval="inverse",yval="zvalue",
      legend="topleft") # x=1/sigma, y=z-value
plot(a,xval="sigma",log="x",yval="pvalue",
      legend="topleft") # x=log(sigma), y=probability
## plot of extrapolation
plot(summary(a),legend="topleft")
## multiple plots
b <- mam15.relltest[1:15] # an object of class "scalebootv"
plot(b) # x=sigma^2, y=psi
```

---

relltest

*RELL Test for Phylogenetic Inference*

---

**Description**

Performs the RELL test for finding the largest item. This calculates AU p-values for each item via the multiscale bootstrap resampling. This is particularly useful for testing tree topologies in phylogenetic analysis.

**Usage**

```
relltest(dat,nb=10000,sa=9^seq(-1,1,length=13),ass=NULL,
         cluster=NULL,nofit=FALSE,models=NULL,seed=100)
```

**Arguments**

dat	a matrix. Row vectors are to be resampled. Each column vector gives score values to be evaluated for an item. For the phylogenetic analysis, $\text{dat}[i, j]$ is the site-wise log-likelihood value at site- $i$ for tree- $j$ , and we are to find the tree with the largest expected value of $\text{sum}(\text{dat}[, j])$ .
nb	Number of replicates for each scale.
sa	Scales in sigma squared ( $\sigma^2$ ).
ass	A list of association vectors for testing edges as well as trees. If $\text{ass}=\text{NULL}$ , then only the results for trees are returned.
cluster	<b>snow</b> cluster object which may be generated by function <code>makeCluster</code> .

nofit	logical. Passed to <code>sbfit</code> .
models	character vectors. Passed to <code>sbfit</code> .
seed	If non NULL, then a random seed is set. Specifying a seed is particularly important when <code>cluster</code> is non NULL, in this case <code>seed + seq(along=cluster)</code> are set to cluster nodes.

## Details

`relltest` performs the resampling of estimated log-likelihoods (RELL) method of Kishino et al. (1990). For resampling indices stored in a vector `i`, the resampled log-likelihood for a tree-`j` is approximately calculated by `sum(dat[i, j])`. This approximation avoids time-consuming recalculation of the maximum likelihood estimates of tree parameters, which are to be calculated by an external phylogenetic software such as PAML as described in [mam15](#). In the implementation of `relltest`, the resampled log-likelihood is calculated by `sum(dat[i, j])*nrow(dat)/length(i)` so that the statistic is comparable to the case when  $n' = n$ .

`relltest` first calls `scaleboot` internally for multiscale bootstrap resampling, and then `scaleboot` calls `sbfit` for fitting models to the bootstrap probabilities. The AU p-values (named "k.3") produced by the summary method are improvements of the third-order p-values calculated by CONSEL software (Shimodaira and Hasegawa 2001). In addition, `relltest` calls `scaleboot` with `sa=1` for calculating p-values via the Shimodaira-Hasegawa test (SH-test) of Shimodaira and Hasegawa (1999).

See [mam15](#) for details through an example.

## Value

`relltest` returns an object of class "`relltest`" that is inherited from the class "`scalebootv`" by adding two extra components called "`stat`" and "`shtest`". "`stat`" is a vector of the test statistics from the SH-test (i.e., the log-likelihood differences), and "`shtest`" is a list with two components: "`pv`", a vector of SH-test p-values, and "`pe`", a vector of standard errors of the p-values. The results of multiscale bootstrap resampling are stored in the "`scalebootv`" components returned by a call to `sbfit`.

## Author(s)

Hidetoshi Shimodaira

## References

- Kishino, H., Miyata, T. and Hasegawa, M. (1990). Maximum likelihood inference of protein phylogeny and the origin of chloroplasts., *J. Mol. Evol.*, 30, 151-160.
- Shimodaira, H. and Hasegawa, M. (1999). Multiple comparisons of log-likelihoods with applications to phylogenetic inference, *Molecular Biology and Evolution*, 16, 1114-1116.
- Shimodaira, H. and Hasegawa, M. (2001). CONSEL: for assessing the confidence of phylogenetic tree selection, *Bioinformatics*, 17, 1246-1247 (software is available from <http://www.is.titech.ac.jp/~shimo/prog/consel/>).
- Luke Tierney, A. J. Rossini, Na Li and H. Sevcikova. snow: Simple Network of Workstations. R package version 0.2-1.

**See Also**

[sbfite](#), [scaleboot](#), [mam15](#).

**Examples**

```
## Not run:
## An example from data(mam15).
## It may take 20 mins to run relltest below.
mam15.mt <- read.mt("mam15.mt") # site-wise log-likelihoods
mam15.trees <- relltest(mam15.mt) # resampling and fitting
summary(mam15.trees) # AU p-values

## End(Not run)
```

---

sbaic

*Akaike's Information Criterion*

---

**Description**

Extract or modify the AIC values for models.

**Usage**

```
sbaic(x,...)
## S3 method for class 'scaleboot'
sbaic(x,k,...)
## S3 method for class 'scalebootv'
sbaic(x,...)

sbaic(x) <- value
## S3 replacement method for class 'scaleboot'
sbaic(x) <- value
## S3 replacement method for class 'scalebootv'
sbaic(x) <- value
```

**Arguments**

x	an object used to select a method.
k	numeric, the penalty per parameter to be used.
value	numeric vector of AIC values for models.
...	further arguments passed to and from other methods.

**Details**

sbaic can be used to modify the aic components for models in x as shown in the examples below.

**Value**

For an object of class "scaleboot", sbaic returns a numeric vector of AIC values for models. If k is missing, then the aic components in the fi vector of x are returned. If k is specified,  $rss - k \cdot df$  is calculated for each model. For the usual AIC,  $k=2$ . For the BIC (Schwarz's Bayesian information criterion),  $k = \log(\sum(x\$nb))$ .

**Author(s)**

Hidetoshi Shimodaira

**References**

Sakamoto, Y., Ishiguro, M., and Kitagawa G. (1986). *Akaike Information Criterion Statistics*. D. Reidel Publishing Company.

**See Also**

[sbfit](#).

**Examples**

```
data(mam15)
a <- mam15.relltest[["t4"]] # an object of class "scaleboot"
sbaic(a) # print AIC for models
sbaic(a,k=log(sum(a$nb))) # print BIC for models
sbaic(a) <- sbaic(a,k=log(sum(a$nb))) # set BIC
sbaic(a) # print BIC for models
```

---

sbconf

*Bootstrap Confidence Intervals*

---

**Description**

A confidence interval for a scalar parameter is obtained by inverting the approximately unbiased p-value. This function is very slow, and it is currently experimental.

**Usage**

```
sbconf(x, ...)

## Default S3 method:
sbconf(x,sa, probs=c(0.05,0.95), model="poly.2",
       k=2,s=1,sp=-1, cluster=NULL,...)

## S3 method for class 'sbconf'
sbconf(x, probs=x$probs,model=x$model,
       k=x$k,s=x$s,sp=x$sp, nofit=FALSE, ...)
```

```
## S3 method for class 'sbconf'
plot(x,model=x$model,k=x$k,s=x$s,sp=x$sp,
     models = attr(x$fits,"models"), log.xy = "",
     xlab="test statistic",ylab=NULL, type.plot = c("p","l","b"),
     yval=c("aic","zvalue","pvalue"), sd=2,add=FALSE, col=1:6,
     pch=NULL,lty=1:5,lwd=par("lwd"), mk.col=col[1],
     mk.lwd=lwd[1], mk.lty=lty[1], ...)
```

## Arguments

x	an object used to select a method. For <code>sbconf.default</code> , x is a list vector of size <code>length{sa}</code> with each element being a vector of bootstrap replicates of a statistic or a list vector of a scalar component.
...	further arguments passed to or from other methods.
sa	vector of scales in sigma squared ( $\sigma^2$ ).
probs	a vector of probabilities at which p-values are inverted.
model	a character to specify a model for an AU p-value. This should be included in <code>soptions("models")</code> , for which model fitting is made internally.
k	a numeric to specify an order of AU p-value.
s	$\sigma_0^2$
sp	$\sigma_p^2$
cluster	<b>snow</b> cluster object which may be generated by function <code>makeCluster</code> .
nofit	logical. No further calls to <code>sbfit</code> are made.
models	AIC values are plotted for these models.
log.xy	character string to be passed to <code>plot.default</code> .
xlab	a label for the x axis.
ylab	a label for the y axis.
type.plot	a character to be passed to <code>plot.default</code> .
yval	determines y-axis. "aic" for AIC values of models, "zvalue" for AU corrected z-values, and "pvalue" for AU corrected p-values.
sd	If positive, draws curves $\pm$ sd*standard error for z-values and p-values.
add	logical. Should not the frame be drawn?
col	vector of colors of plots.
pch	vector of pch's of plots.
lty	vector of lty's of plots.
lwd	numeric of lwd of plots.
mk.col	color for crosses drawn at probs.
mk.lwd	lwd for crosses drawn at probs.
mk.lty	lty for crosses drawn at probs.

## Details

Let  $x[[i]]$  be a vector of bootstrap replicates for a statistic with scale  $sa[i]$ . For a threshold value  $y$ , the bootstrap probability is  $bp[i]=\text{sum}(x[[i]]<y)/\text{length}(x[[i]])$ . `sbconf` computes  $bp$  for several  $y$  values, and finds a value  $y$  at which the AU p-value, given by `sbfit`, equals a probability value specified in `probs`. In this manner, AU p-values are inverted to obtain bootstrap confidence intervals.

See the examples below for details.

## Value

`sbconf` method returns an object of class "sbconf".

The `print` method for an object of class "sbconf" prints the confidence intervals.

## Author(s)

Hidetoshi Shimodaira

## See Also

[scaleboot](#).

## Examples

```
## Not run:
## An example to calculate confidence intervals
## The test statistic is that for "t4" in data(mam15)
## In the following, we used 40 cpu's.
##
library(snow)
library(scaleboot)
cl <- makeCluster(40)
data(mam15)

## Definition of a test statistic of interest.
## "myfun" returns the maximum difference of log-likelihood value
## for a tree named a.
myfun <- function(x,w,a) maxdif(wsumrow(x,w))[[a]]
maxdif <- function(x) {
  i1 <- which.max(x) # the largest element
  x <- -x + x[i1]
  x[i1] <- -min(x[-i1]) # the second largest value
  x
}
wsumrow <- function(x,w) {
  apply(w*x,2,sum)*nrow(x)/sum(w)
}
clusterExport(cl,c("maxdif","wsumrow"))

## multiscale bootstrap parameters
nb <- 10000
```

```

sa <- 10^seq(-2,2,length=13)

## Compute multiscale bootstrap replicates
## (It took 80 secs using 40 cpu's)
sim <- scaleboot(mam15.mt,nb,sa,myfun,"t4",count=FALSE,
                cluster=cl,onlyboot=TRUE,
                names.hp=na,nofit=nofit,models=models)

## Modify option "probs0" to a fine grid with 400 points
## default: 0.001 0.010 0.100 0.900 0.990 0.999
## NOTE: This modification is useful only when cl != NULL,
##       in which case calls to sbfit for the grid points
##       are made in parallel, although iterations seen later
##       are made sequentially.
sboptions("probs0",pnorm(seq(qnorm(0.001),qnorm(0.999),length=400)))

## Calculate bootstrap confidence intervals using "k.1" p-value.
## (It took 70 secs using 40 cpu's)
## First, sbfit is applied to bp's determined by option "probs0"
## Then, additional fitting is made only twice for iteration.
## p[1]=0.05 iter=1 t=4.342723 e=0.0003473446 r=0.0301812
## p[2]=0.95 iter=1 t=42.76558 e=0.002572495 r=0.1896809
conf1 <- sbconf(sim$stat,sim$sa,model="sing.3",k=1,cluster=cl)

## The confidence interval with "k.1" is printed as
##      0.05      0.95
## 4.342723 42.765582
conf1

## Calculate bootstrap confidence intervals
##                               using "k.2" and "k.3" p-values.
## (It took only 10 secs)
## p[1]=0.05 iter=1 t=-2.974480 e=0.003729190 r=0.04725755
## p[2]=0.95 iter=1 t=39.51767 e=0.001030929 r=0.06141937
##      0.05      0.95
## -2.974480 39.517671
conf2 <- sbconf(conf1,model="sing.3",k=2)
conf2
## p[1]=0.05 iter=1 t=-3.810157 e=0.01068678 r=0.08793868
## p[2]=0.95 iter=1 t=39.32669 e=0.001711107 r=0.09464663
##      0.05      0.95
## -3.810157 39.326686
conf3 <- sbconf(conf2,model="sing.3",k=3)
conf3

## plot diagnostics
plot(conf3) # AIC values for models v.s. test statistic value
plot(conf3,yval="zval",type="l") # corrected "k.3" z-value

stopCluster(cl)

## End(Not run)

```

**Description**

sbfit is used to fit parametric models to multiscale bootstrap probabilities by the maximum likelihood method.

**Usage**

```
sbfit(x, ...)

## Default S3 method:
sbfit(x,nb,sa,models=NULL,nofit=FALSE,...)

## S3 method for class 'matrix'
sbfit(x,nb,sa,models=NULL,names.hp=rownames(x),
      nofit=FALSE,cluster=NULL,...)

## S3 method for class 'data.frame'
sbfit(x,...)

## S3 method for class 'scaleboot'
sbfit(x,models=names(x$fi),...)

## S3 method for class 'scalebootv'
sbfit(x,models=attr(x,"models"),...)

## S3 method for class 'scaleboot'
print(x,sort.by=c("aic","none"),...)

## S3 method for class 'scalebootv'
print(x,...)
```

**Arguments**

x	an object used to select a method. For <code>sbfit.default</code> , x is denoted by <code>nb</code> and is a vector of bootstrap probabilities for a hypothesis. For <code>sbfit.matrix</code> , x is denoted by <code>bps</code> and is a matrix with row vectors of <code>bp</code> for several hypotheses.
nb	vector of numbers of bootstrap replicates. A short vector (or scalar) is cyclically extended to match the size of <code>bp</code> .
sa	vector of scales in sigma squared ( $\sigma^2$ ). Should be the same size as <code>bp</code> .
models	character vector of model names. Valid model names are <code>poly.m</code> for $m \geq 1$ and <code>sing.m</code> for $m \geq 3$ . The default is set by <code>sboptions()\$models</code> , whose default is <code>c("poly.1","poly.2","poly.3","sing.3","sphe.3")</code> . If <code>models</code> is an integer value, <code>sbmodelnames(m=models)</code> is used.

<code>nofit</code>	logical. If TRUE, fitting is not performed.
<code>names.hp</code>	character vector of hypotheses names.
<code>cluster</code>	<b>snow</b> cluster object which may be generated by function <code>makeCluster</code> .
<code>sort.by</code>	sort key.
<code>...</code>	further arguments passed to and from other methods.

## Details

`sbfit.default` fits parametric models to `bp` by maximizing the log-likelihood value of a binomial model. A set of multiscale bootstrap resampling should be performed before a call to `sbfit` for preparing `bp`, where `bp[i]` is a bootstrap probability of a hypothesis calculated with a number of bootstrap replicates `nb[i]` and a scale  $\sigma^2=sa[i]$ . The scale is defined as  $\sigma^2 = n/n'$ , where  $n$  is the sample size of data, and  $n'$  is the sample size of replicated data for bootstrap resampling.

Each model specifies a  $\psi(\beta, s) = \psi(\sigma^2 | \beta)$  function with a parameter vector  $\beta$ . The model may describe how the bootstrap probability changes along the scale. Let `cnt[i]=bp[i]*nb[i]` be the frequency indicating how many times the hypothesis of interest is observed in bootstrap replicates at scale `sa[i]`. Then we assume that `cnt[i]` is binomially distributed with number of trials `nb[i]` and success probability  $1-pnorm(\psi(\beta, s=sa[i])/sqrt(sa[i]))$ . Currently, `sbpsi.poly` and `sbpsi.sing` are available as  $\psi$  functions. The estimated model parameters are accessed by the `coef.scaleboot` method.

The model fitting is performed in the order specified in `models`, and the initial values for numerical optimization of the likelihood function are prepared by using previously estimated model parameters. Thus, "poly.(m-1)" should be specified before "poly.m", and "poly.(m-1)" and "sing.(m-1)" should be specified before "sing.m".

`sbfit.matrix` calls `sbfit.default` repeatedly, once for each row vector `bp` of the matrix `bps`. Parallel computing is performed when `cluster` is non NULL.

`sbfit.scaleboot` calls `sbfit.default` with `bp`, `nb`, and `sa` components in `x` object for refitting by giving another `models` argument. It discards the previous result of fitting, and recomputes the model parameters.

`sbfit.scalebootv` calls `sbfit.matrix` with the `bps`, `nb`, and `sa` components in the attributes of `x`.

## Value

`sbfit.default` and `sbfit.scaleboot` return an object of class "scaleboot", and `sbfit.matrix` and `sbfit.scalebootv` return an object of class "scalebootv".

An object of class "scaleboot" is a list containing at least the following components:

<code>bp</code>	the vector of bootstrap probabilities used.
<code>nb</code>	the <code>rep(nb, length=length(bp))</code> used.
<code>sa</code>	the <code>sa</code> used.
<code>fi</code>	list vector of fitted results for <code>models</code> used. Each list consists of components "par" (estimated parameter), "mag" (magnification factor for "par" to make the actual parameter vector $\beta = \text{par} * \text{mag}$ ), "value" (maximum log-likelihood),

"hessian" (hessian matrix), "var" (variance estimate of "par"), "mask" (logical vector indicating parameter elements which are not at boundaries), "init" (initial values used for optimization), "psi" (psi function name of the model), "df" (degrees of freedom), "rss" (equivalent to the residual sum of squares, but actually defined as  $2*(lik0-lik)$  where lik0 and lik are the log-likelihood function of the non-restricted model and the model of interest, respectively), "pfit" (p-value for "rss"), "aic" (aic value of the model relative to the non-restricted model).

An object of class "scalebootv" is a vector of "scaleboot" objects, and in addition, it has attributes "models", "bps", "nb", and "sa".

### Author(s)

Hidetoshi Shimodaira <shimo@is.titech.ac.jp>

### References

Shimodaira, H. (2002). An approximately unbiased test of phylogenetic tree selection, *Systematic Biology*, 51, 492-508.

Shimodaira, H. (2004). Approximately unbiased tests of regions using multistep-multiscale bootstrap resampling, *Annals of Statistics*, 32, 2616-2641.

Shimodaira, H. (2008). Testing Regions with Nonsmooth Boundaries via Multiscale Bootstrap, *Journal of Statistical Planning and Inference*, 138, 1227-1241. (<http://dx.doi.org/10.1016/j.jspi.2007.04.001>).

### See Also

[sbpsi](#), [summary.scaleboot](#), [plot.scaleboot](#), [coef.scaleboot](#), [sbaic](#).

### Examples

```
## Testing a hypothesis
## Examples of fitting models to a vector of bp's
## mam15.relltest$t4 of data(mam15), but
## using a different set of scales (sigma^2 values).
## In the below, sigma^2 ranges 0.01 to 100 in sa[i]
## This very large range is only for illustration.
## Typically, the range around 0.1 to 10
## is recommended for much better model fitting.
## In other examples, we have used
## sa = 9^seq(-1,1,length=13).

cnt <- c(0,0,0,0,6,220,1464,3565,5430,6477,6754,
        6687,5961) # observed frequencies at scales
nb <- 100000 # number of replicates at each scale
bp <- cnt/nb # bootstrap probabilities (bp's)
sa <- 10^seq(-2,2,length=13) # scales (sigma squared)
## model fitting to bp's
f <- sbfit(bp,nb,sa) # model fitting ("scaleboot" object)
f # print the result of fitting
```

```

plot(f,legend="topleft") # observed bp's and fitted curves
## approximately unbiased p-values
summary(f) # calculate and print p-values
## refitting with models up to "poly.4" and "sing.4"
f <- sbfit(f,models=1:4)
f # print the result of fitting
plot(f,legend="topleft") # observed bp's and fitted curves
summary(f) # calculate and print p-values

## Testing multiple hypotheses (only two here)
## Examples of fitting models to vectors of bp's
## mam15.relltest[c("t1,t2")]
cnt1 <- c(85831,81087,76823,72706,67946,62685,57576,51682,
          45887,41028,35538,31232,27832) # cnt for "t1"
cnt2 <- c(2,13,100,376,975,2145,3682,5337,7219,8559,
          10069,10910,11455) # cnt for "t2"
cnts <- rbind(cnt1,cnt2)
nb <- 100000 # number of replicates at each scale
bps <- cnts/nb # row vectors are bp's
sa <- 9^seq(-1,1,length=13) # scales (sigma squared)
fv <- sbfit(bps,nb,sa) # returns a "scalebootv" object
fv # print the result of fitting
plot(fv) # multiple plots
summary(fv) # calculate and print p-values

```

---

sboptions

*Options for Multiscale Bootstrap*


---

## Description

To set and examine global options for scaleboot package.

## Usage

```
sboptions(x, value)
```

## Arguments

x	character of an option name.
value	When specified, this value is set.

## Details

Invoking sboptions() with no arguments returns a list with the current values of the options. Otherwise it returns option(s) with name(s) specified by x. When value is specified, it is set to the option named x.

**Author(s)**

Hidetoshi Shimodaira

**Examples**

```

sboptions() # show all the options
sboptions("models") # show the default model names
new.models <- sbmodelnames(m=1:2) # character vector c("poly.1","poly.2")
old.models <- sboptions("models",new.models) # set the new model names
sboptions("models") # show the default model names
sboptions("models",old.models) # set back the default value
sboptions("models") # show the default model names

```

sbpsi

*Model Specification Functions***Description**

sbpsi.poly and sbpsi.sing are  $\psi$  functions to specify a polynomial model and a singular model, respectively.

**Usage**

```

sbpsi.poly(beta,s=1,k=1,sp=-1,lambda=NULL,aux=NULL,check=FALSE)
sbpsi.sing(beta,s=1,k=1,sp=-1,lambda=NULL,aux=NULL,check=FALSE)
sbpsi.sphe(beta,s=1,k=1,sp=-1,lambda=NULL,aux=NULL,check=FALSE)
sbpsi.generic(beta,s=1,k=1,sp=-1,lambda=NULL,aux=NULL,check=FALSE,zfun,eps=0.01)
sbmodelnames(m=1:3,one.sided=TRUE,two.sided=FALSE,rev.sided=FALSE,poly,sing,poa,pob,sia,sib,sphe)

```

**Arguments**

beta	numeric vector of parameters; $\beta_0=\text{beta}[1]$ , $\beta_1=\text{beta}[2]$ ,... $\beta_{m-1}=\text{beta}[m]$ , where $m$ is the number of parameters.
s	$\sigma_0^2$ .
k	numeric to specify the order of derivatives.
sp	$\sigma_p^2$ .
lambda	a numeric of specifying the type of p-values; Bayesian (lambda=0) Frequentist (lambda=1).
aux	auxiliary parameter. Currently not used.
check	logical for boundary check.

zfun	z-value function with (s,beta) as parameters.
eps	delta for numerical computation of derivatives.
m	numeric vector to specify the numbers of parameters.
one.sided	logical to include poly and sing models.
two.sided	logical to include poa and sia models.
rev.sided	logical to include pob and sib models.
poly	maximum number of parameters in poly models.
sing	maximum number of parameters in sing models.
sphe	maximum number of parameters in sphe models.
poa	maximum number of parameters in poa models.
pob	maximum number of parameters in pob models.
sia	maximum number of parameters in sia models.
sib	maximum number of parameters in sib models.

## Details

For  $k = 1$ , the sbpsi functions return their  $\psi$  function values at  $\sigma^2 = \sigma_0^2$ . Currently, four types of sbpsi functions are implemented. `sbpsi.poly` defines the polynomial model;

$$\psi(\sigma^2|\beta) = \sum_{j=0}^{m-1} \beta_j \sigma^{2j}$$

for  $m \geq 1$ . `sbpsi.sing` defines the singular model;

$$\psi(\sigma^2|\beta) = \beta_0 + \sum_{j=1}^{m-2} \frac{\beta_j \sigma^{2j}}{1 + \beta_{m-1}(\sigma - 1)}$$

for  $m \geq 3$  and  $0 \leq \beta_{m-1} \leq 1$ . `sbpsi.sphe` defines the spherical model; currently the number of parameters must be  $m=3$ . `sbpsi.generic` is to calculate psi value and extrapolation from a given z-function.

For  $k > 1$ , the sbpsi functions return values extrapolated at  $\sigma^2 = \sigma_p^2$  using derivatives up to order  $k - 1$  evaluated at  $\sigma^2 = \sigma_0^2$ ;

$$q_k = \sum_{j=0}^{k-1} \frac{(\sigma_p^2 - \sigma_0^2)^j}{j!} \left. \frac{d^j \psi(x|\beta)}{dx^j} \right|_{\sigma_0^2},$$

which reduces to  $\psi(\sigma_0^2|\beta)$  for  $k = 1$ . In the [summary.scaleboot](#), the AU p-values are defined by  $p_k = 1 - \Phi(q_k)$  for  $k \geq 1$ .

## Value

`sbpsi.poly` and `sbpsi.sing` are examples of a sbpsi function; users can develop their own sbpsi functions for better model fitting by preparing `sbpsi.foo` and `sbini.foo` functions for model foo. If `check=FALSE`, a sbpsi function returns the  $\psi$  function value or the extrapolation value. If

check=TRUE, a sbpsi function returns NULL when all the elements of beta are included in the their valid intervals. Otherwise, a sbpsi function returns a list with components beta for the parameter value being modified to be on a boundary of the interval and mask, a logical vector indicating which elements are not on the boundary.

sbmodelnames returns a character vector of model names.

### Author(s)

Hidetoshi Shimodaira

### See Also

[sbfitt](#).

---

sbpval

*Extract P-values*

---

### Description

sbpval extracts p-values from "summary.scaleboot" or "summary.scalebootv" objects.

### Usage

```
sbpval(x, ...)
```

```
## S3 method for class 'summary.scaleboot'
sbpval(x, sd=FALSE, select=c("average", "best", "all"), ...)
```

```
## S3 method for class 'summary.scalebootv'
sbpval(x, ...)
```

### Arguments

x	an object used to select a method.
sd	logical. Should standard errors be returned as well?
select	character. If "average" or "best", only the p-values of corresponding models are returned. If "all", then p-values of all the models are returned.
...	further arguments passed to or from other methods.

### Details

This method is used only to extract previously calculated p-values from the summary object.

**Value**

The sbpval method for the class "summary.scaleboot" returns a matrix of p-values. Each row is a vector of  $p_k$  for  $k$  as specified in the summary method. If sd=TRUE, then it returns a list with two components: estimate for the p-values and sd for its standard errors.

The sbpval method for the class "summary.scalebootv" returns a list vector with each component obtained by applying sbpval to each "scaleboot" object.

**Author(s)**

Hidetoshi Shimodaira

**See Also**

[summary.scaleboot.](#)

**Examples**

```
data(mam15)
a <- mam15.relltest[["t4"]] # an object of class "scaleboot"
b <- summary(a) # calculate p-values
b # print the p-values
sbpval(b) # extract a vector of p-values which are averaged by Akaike weights.
sbpval(b,sd=TRUE) # with sd
sbpval(b,select="all") # extract a matrix of p-values
sbpval(b,select="all",sd=TRUE) # with sd
```

---

scaleboot

*Multiscale Bootstrap Resampling*

---

**Description**

Performs multiscale bootstrap resampling for a specified statistic.

**Usage**

```
scaleboot(dat,nb,sa,fun,parm=NULL,count=TRUE,weight=TRUE,
          cluster=NULL,onlyboot=FALSE,seed=NULL,...)
```

```
countw.assmax(x,w,ass)
```

```
countw.shtest(x,w,obs)
```

```
countw.shtestass(x,w,assobs)
```

**Arguments**

dat	data matrix or data-frame. Row vectors are to be resampled.
nb	vector of the numbers of bootstrap replicates.
sa	vector of scales in sigma squared ( $\sigma^2$ ).
fun	function for a statistic.
parm	parameter to be passed to fun above.
count	logical. Should only the accumulative counts be returned? Otherwise, raw statistic vectors are returned.
weight	logical. In fun above, resampling is specified by a weight vector. Otherwise, resampling is specified by a vector of indices.
cluster	<b>snow</b> cluster object which may be generated by function makeCluster.
onlyboot	logical. Should only bootstrap resampling be performed? Otherwise, <a href="#">sbfit</a> or <a href="#">sbconf</a> is called internally.
seed	If non NULL, random seed is set. Specifying a seed is particularly important when cluster is non NULL, in which case seed + seq(along=cluster) are set to cluster nodes.
...	further arguments passed to and from other methods.
x	data matrix or data-frame passed from <a href="#">scaleboot</a> .
w	weight vector for resampling.
ass	a list of association vectors. An example of parm above.
obs	a vector of observed test statistics. An example of parm above.
assobs	a list of ass and obs above. An example of parm above.

**Details**

These functions are used internally by [relltest](#).

scaleboot performs multiscale bootstrap resampling for a statistic defined by fun, which should be one of the two possible forms fun(x,w,parm) and fun(x,i,parm). The former is used when weight=TRUE, and the weight vector w is generated by a multinomial distribution. The latter is used when weight=FALSE, and the index vector i is generated by resampling  $n'$  elements from  $\{1, \dots, n\}$ . When count=TRUE, fun should return a logical, or a vector of logicals.

Examples of fun(x,w,parm) are countw.assmax for AU p-values, countw.shstest for SH-test of trees, and countw.shstestass for SH-test of both trees and edges. The definitions are given below.

```
countw.assmax <- function(x,w,ass) {
  y <- maxdif(wsumrow(x,w)) <= 0 # countw.max
  if(is.null(ass)) y
  else {
    z <- vector("logical",length(ass))
    for(i in seq(along=ass)) z[i] <- any(y[ass[[i]])]
    z
  }
}
```

```

countw.shtest <- function(x,w,obs)  maxdif(wsumrow(x,w)) >= obs

countw.shtestass <- function(x,w,assobs)
  unlist(assmaxdif(wsumrow(x,w),assobs$ass)) >= assobs$obs

### weighted sum of row vectors
##
## x = matrix (array of row vectors)
## w = weight vector (for rows)
##
wsumrow <- function(x,w) {
  apply(w*x,2,sum)*nrow(x)/sum(w)
}

### calc max diff
##
## y[i] := max_{j neq i} x[j] - x[i]
##
maxdif <- function(x) {
  i1 <- which.max(x) # the largest element
  x <- -x + x[i1]
  x[i1] <- -min(x[-i1]) # the second largest value
  x
}

### calc assmaxdif
##
## y[[i]][j] := max_{k neq ass[[i]]} x[k] - x[ass[[i]][j]]
##
assmaxdif <- function(x,a) {
  y <- vector("list",length(a))
  names(y) <- names(a)
  for(i in seq(along=a)) y[[i]] <- max(x[-a[[i]]) - x[a[[i]])
  y
}

```

When `count=TRUE`, the summation of outputs from `fun` is calculated. This gives the frequencies for how many times the hypotheses are supported by the bootstrap replicates.

### Value

If `onlyboot=TRUE`, then a list of raw results from the multiscale bootstrap resampling is returned. The components are "stat" for list vectors of outputs from `fun` (only when `count=FALSE`), "bps" for a matrix of multiscale bootstrap probabilities (only when `count=FALSE`), "nb" for the number of bootstrap replicates used, and "sa" for the scales used. Note that scales are redefined by `sa <- nsize/round(nsize/sa)`, where `nsize` is the sample size.

If `onlyboot=FALSE`, then the result of a call to `sbfit` is returned when `count=TRUE`, otherwise the result of `sbconf` is returned when `count=FALSE`.

**Author(s)**

Hidetoshi Shimodaira

**See Also**[sbfit](#), [relltest](#).**Examples**

```
## Not run:
## a line from the definition of relltest
scaleboot(dat,nb,sa,countw.assmax,ass,cluster=cluster,
          names.hp=na,nofit=nofit,models=models,seed=seed)

## two lines from rell.shtest (internal function)
scaleboot(z,nb,1,countw.shtest,tobs,cluster=cluster,
          onlyboot=TRUE,seed=seed)
scaleboot(z,nb,1,countw.shtestass,pa,cluster=cluster,
          onlyboot=TRUE,seed=seed)

## End(Not run)
```

---

summary.scaleboot      *P-value Calculation for Multiscale Bootstrap*

---

**Description**

summary method for class "scaleboot" and "scalebootv".

**Usage**

```
## S3 method for class 'scaleboot'
summary(object,models=names(object$fi),k=3,s=1,sp=-1,
        type=c("Frequentist","Bayesian"),...)

## S3 method for class 'scalebootv'
summary(object,models=attr(object,"models"),k=3,type="Frequentist",...)

## S3 method for class 'summary.scaleboot'
print(x,sort.by=c("aic","none"),verbose=FALSE,...)

## S3 method for class 'summary.scalebootv'
print(x,select="average",sort.by=NULL,nochisq=TRUE,...)
```

**Arguments**

object	an object used to select a method.
models	character vector of model names. If numeric, <code>names(object\$fi)[models]</code> is used for each "scaleboot" object.
k	numeric vector of $k$ for calculating p-values.
s	$\sigma_0^2$
sp	$\sigma_p^2$
type	If numeric, it is passed to sbpsi functions as lambda to specify p-value type. If "Frequentist" or "Bayesian", then equivalent to specifying <code>lambda = 1</code> or <code>0</code> , respectively.
select	character of model name (such as "poly.3") or one of "average" and "best". If "average" or "best", then the averaging by Akaike weights or the best model is used, respectively.
x	object.
sort.by	sort key.
verbose	logical.
nochisq	logical.
...	further arguments passed to and from other methods.

**Details**

For each model, a class of approximately unbiased p-values, indexed by  $k = 1, 2, \dots$ , is calculated. The p-values are named `k.1`, `k.2`, ..., where  $k = 1$  (`k.1`) corresponds to the ordinary bootstrap probability, and  $k = 2$  (`k.2`) corresponds to the third-order accurate p-value of Shimodaira (2002). As the  $k$  value increases, the bias of testing decreases, although the p-value becomes less stable numerically and the monotonicity of rejection regions becomes worse. Typically,  $k = 3$  provides a reasonable compromise. The `sbpval` method is available to extract p-values from the "summary.scaleboot" object.

The p-value is defined as

$$p_k = 1 - \Phi \left( \sum_{j=0}^{k-1} \frac{(\sigma_p^2 - \sigma_0^2)^j}{j!} \frac{d^j \psi(x|\beta)}{dx^j} \Big|_{\sigma_0^2} \right),$$

where  $\psi(\sigma^2|\beta)$  is the model specification function,  $\sigma_0^2$  is the evaluation point for the Taylor series, and  $\sigma_p^2$  is an additional parameter. Typically, we do not change the default values  $\sigma_0^2 = 1$  and  $\sigma_p^2 = -1$ .

The p-values are justified only for good fitting models. By default, the model which minimizes the AIC value is selected. We can modify the AIC value by using the `sbaic` function. We also diagnose the fitting by using the `plot` method.

**Value**

summary.scaleboot returns an object of the class "summary.scaleboot", which is inherited from the class "scaleboot". It is a list containing all the components of class "scaleboot" and the following components:

pv	matrix of p-values of size length(models) * length(k) with elements $p_k$ .
pe	matrix of standard errors of p-values.
best	a list consisting of components model for the best fitting model name, aic for its AIC value, pv for a vector of p-values, and pe for a vector of standard errors.
parex	a list of components k, s, and sp.

**Author(s)**

Hidetoshi Shimodaira

**See Also**

[sbfit](#), [sbpsi](#), [sbpval](#), [sbaic](#).

**Examples**

```
data(mam15)
## For a single hypothesis
a <- mam15.relltest[["t4"]] # an object of class "scaleboot"
summary(a) # calculate and print p-values (k=3)
summary(a,k=2) # calculate and print p-values (k=2)
summary(a,k=1:4) # up to "k.4" p-value.

## For multiple hypotheses
b <- mam15.relltest[1:15] # an object of class "scalebootv"
summary(b) # calculate and print p-values (k=3)
summary(b,k=1:4) # up to "k.4" p-value.
```

# Index

- \*Topic **datasets**
  - lung73, 5
  - mam15, 8
- \*Topic **environment**
  - sboptions, 27
- \*Topic **file**
  - interface, 3
- \*Topic **hplot**
  - plot.scaleboot, 15
- \*Topic **models**
  - coef, 2
  - plot.scaleboot, 15
  - sbaic, 19
  - sbfit, 24
  - sboptions, 27
  - sbpsi, 28
  - sbpval, 30
  - scaleboot-package, 2
  - summary.scaleboot, 34
- \*Topic **nonparametric**
  - relltest, 17
  - sbconf, 20
  - sbfit, 24
  - sbpval, 30
  - scaleboot, 31
  - scaleboot-package, 2
  - summary.scaleboot, 34
- \*Topic **package**
  - scaleboot-package, 2
- coef, 2
- coef.scaleboot, 25, 26
- countw.assmax (scaleboot), 31
- countw.shtest (scaleboot), 31
- countw.shtestass (scaleboot), 31
- interface, 3
- lines.scaleboot (plot.scaleboot), 15
- lung73, 5
- mam15, 8, 18, 19
- plot.sbconf (sbconf), 20
- plot.scaleboot, 15, 26
- plot.scalebootv (plot.scaleboot), 15
- plot.summary.scaleboot (plot.scaleboot), 15
- plot.summary.scalebootv (plot.scaleboot), 15
- print.sbconf (sbconf), 20
- print.scaleboot (sbfit), 24
- print.scalebootv (sbfit), 24
- print.summary.scaleboot (summary.scaleboot), 34
- print.summary.scalebootv (summary.scaleboot), 34
- read.ass, 14
- read.ass (interface), 3
- read.cnt (interface), 3
- read.mt, 14
- read.mt (interface), 3
- relltest, 5, 10, 14, 17, 32, 34
- sbaic, 19, 26, 36
- sbaic<- (sbaic), 19
- sbconf, 20, 32, 33
- sbfit, 3, 17–20, 24, 30, 32–34, 36
- sbfit.pvclust, 7
- sbfit.pvclust (interface), 3
- sblegend (plot.scaleboot), 15
- sbmodelnames (sbpsi), 28
- sboptions, 27
- sbpsi, 26, 28, 36
- sbpval, 30, 35, 36
- sbpvclust, 7
- sbpvclust (interface), 3
- scaleboot, 10, 18, 19, 22, 31, 32
- scaleboot-package, 2
- summary.relltest (relltest), 17

`summary.scaleboot`, 26, 29, 31, 34  
`summary.scalebootv`, 14  
`summary.scalebootv (summary.scaleboot)`,  
34