MULTISCALE BOOTSTRAP USING SCALEBOOT PACKAGE

HIDETOSHI SHIMODAIRA

1. Introduction

The `scaleboot` is an add-on package of R. This is for calculating approximately unbiased (AU) p-values for a general problem from a set of multiscale bootstrap probabilities (BPs). Scaling is equivalent to changing the sample size of dataset in bootstrap resampling. We compute BPs at several scales, from which a very accurate p-value is calculated (Shimodaira 2002). This multiscale bootstrap method has been implemented in `CONSEL` software (Shimodaira and Hasegawa 2001) for phylogenetic inference and `pvclust` package (Suzuki and Shimodaira 2006) for hierarchical clustering. The thrust of `scaleboot` package is to calculate an improved version of AU p-values which are justified even for hypotheses with nonsmooth boundaries (Shimodaira 2006).

The basic usage of this package is illustrated in a simple example below. Then real applications in hierarchical clustering and phylogenetic inference are shown later.

2. Install

`scaleboot` is easily installed from CRAN online. Windows users may use the pull-down menu for install and just choose "scaleboot". Otherwise, run R on your computer and type

```
> install.packages("scaleboot")
```

You can also download the package file from the URL below, and install manually.

`http://www.is.titech.ac.jp/~shimo/prog/scaleboot/`

3. Simple Example

3.1. **Simulation Data.** We first generate a simulation dataset.

```
> simdata <- function(n, y, sd) {
+     m <- length(y)
+     x <- matrix(rnorm(m * n, 0, sd), m, n)
+     t(x + (y - apply(x, 1, mean)))
+ }
> X <- simdata(100, c(0, 1, 1, 1, 1, 1, 1, 1, 1, 1), 10)
> round(X[1:3, ], 3)
       [,1]  [,2]   [,3]   [,4]   [,5]   [,6]   [,7]  [,8]    [,9]  [,10]
[1,] -6.558 0.925  0.312 10.095  2.894  3.519 -4.529 7.589  -6.712 -2.012
[2,] -0.638 0.573 -0.915  8.625  2.958  0.040 -2.599 5.552  -7.598 24.689
[3,] -5.917 7.251  3.721  8.961 -6.419 -4.051 -5.913 2.753 -10.037  4.057
> y <- apply(X, 2, mean)
> round(y, 3)
 [1] 0 1 1 1 1 1 1 1 1 1
```

This package vignette is included in `scaleboot` available from CRAN. The source file is usesb.Rnw (2006-10-30 11:16:44 shimo).

The matrix $X = (x_{ij})$ above is of size $n \times m$ with $n = 100$, $m = 10$. We consider that X is a dataset of sample size n, and rows $x_i = (x_{11}, \ldots, x_{im})$, $i = 1, \ldots, n$, are observations of a random vector of m dimensions.

3.2. **Null Hypothesis.** Let μ be the unknown population mean of the row vectors. An estimate of μ is the sample average of the rows defined as $y = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Let $f(\mu)$ be a 0/1-valued (or false/true valued) function of μ. The null hypothesis we are going to test is represented as $f(\mu) = 1$. For example, $f(\mu) = 1$ if μ_1 is the largest among $\mu_1, \ldots, \mu_m$, and $f(\mu) = 0$ otherwise. This $f(\mu)$ is implemented as `mc1(mu)` below.

```
> mc1 <- function(x) all(x[1] >= x[-1])
> mc1(y)
[1] FALSE
```

Although $f(y) = 0$ gives a rough idea whether $f(\mu) = 1$, we want to calculate a real number ranging between 0 and 1 which indicates the possibility of $f(\mu) = 1$. This is what `scaleboot` calculates as p-values.

3.3. **Bootstrap Probabilities.** A naive way to calculate p-value is the bootstrap resampling. Let $X^* = (x^*_{ij})$ be a bootstrap sample of X; each row x^*_i is obtained by resampling with replacement from the rows $x_1, \ldots, x_n$. Let n' be the size of the resampling so that X^* is a matrix of size $n' \times m$. The bootstrap replicate is $y^* = \bar{x}^* = \frac{1}{n'}\sum_{i=1}^{n'} x^*_i$. The following code generates a X^* of $n' = n$, and calculates $f(y^*)$. The resampling is made via weight vector `w`; w_i is the number of times that x_i is resampled in X^*.

```
> countw <- function(x, w, fn) {
+     y <- apply(w * x, 2, sum)/sum(w)
+     fn(y)
+ }
> w <- as.vector(rmultinom(1, 100, rep(1, 100)))
> w
  [1] 2 2 2 0 1 1 2 0 2 1 2 0 1 0 2 1 0 1 0 0 0 2 0 1 1 0 1 0 0 0 1 1 1 2 0 1 1
 [38] 2 1 0 1 1 1 0 2 1 1 1 0 0 1 1 0 2 1 0 2 1 1 1 3 1 0 2 1 3 0 0 0 0 1 1 2 1
 [75] 1 0 1 1 3 1 2 1 2 0 2 2 1 0 1 4 1 1 0 1 1 1 2 1 1 1
> countw(X, w, mc1)
[1] FALSE
```

Let B be the number of bootstrap samples we will generate, and $y^*_1, \ldots, y^*_B$ be the bootstrap replicates. Typically, $B = 10,000$. The BP is computed as $\sum_{i=1}^{B} f(y^*_i)/B$, where the ordinary BP uses $n' = n$. Since the first introduction by Felsenstein (1985), it has been widely used as a p-value, but the bias is in fact rather large.

3.4. **P-value Calculation.** `scaleboot` calculates corrected p-values for improving BPs. First load the package by

```
> library(scaleboot)
```

In the below, `sa` specifies the scales, and `nb` specifies B for each scale, so that $10,000 \times 13 = 130,000$ bootstrap samples are generated internally. It takes a few minutes on a pc.

```
> sa <- 9^seq(-1, 1, length = 13)
> nb <- 10000
> X.sb <- scaleboot(X, nb, sa, countw, mc1)
```

The result is shown by

```
> summary(X.sb)
Raw Bootstrap Probability:  0.98 (0.10)

Corrected P-values (percent):
       k.1          k.2          k.3          aic
poly.1 0.07 (0.00)  0.07 (0.00)  0.07 (0.00) 3666.22
poly.2 1.05 (0.05)  6.59 (0.35)  6.59 (0.35)  198.16
poly.3 1.21 (0.05) 15.03 (0.95) 18.37 (1.30)   17.23
sing.3 1.03 (0.04) 17.67 (0.74) 40.71 (1.47)  -18.05

Best Model:  sing.3
```

A class of AU p-values p_k indexed by $k = 1, 2, 3$, are calculated, and they are labelled as `k.1`, `k.2`, and `k.3`. The p-values are shown in percent, and the standard errors are given in parentheses. We should look at the row of `sing.3` as indicated as the best model in terms of AIC, and we can ignore the other rows. $p_1 \approx 1\%$ corresponds to the ordinary BP, and $p_2 \approx 18\%$ corresponds to the AU p-value of Shimodaira (2002). What we recommend to use here is $p_3 \approx 41\%$; this is the AU p-value of Shimodaira (2006). For this particular example, the common practice for calculating p-value is to use the multiple comparisons method. If it applied to y, the p-value is $p = 66\%$, which is rather close to p_3 in our example, whereas p_1 is obviously too small.

3.5. **Internal Steps.** We consider the following three steps (i)-(iii). Internally, `scaleboot` function (i) performs the multiscale bootstrap, and (ii) estimates coefficients for candidate models. Then the `summary` method (iii) calculates the corrected p-values. These steps are explained below.

The results of steps (i) and (ii) are shown by

```
> X.sb
Multiscale Bootstrap Probabilities (percent):
1    2    3    4    5    6    7    8    9    10   11   12   13
0.00 0.01 0.05 0.14 0.33 0.66 0.98 1.55 2.16 2.61 3.39 3.92 4.61

Numbers of Bootstrap Replicates:
1     2     3     4     5     6     7     8     9     10    11    12    13
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000

Scales (Sigma Squared):
1      2      3      4      5      6      7 8     9     10   11    12   13
0.1111 0.1603 0.2309 0.3333 0.4808 0.6944 1 1.449 2.083 3.03 4.348 6.25 9.091

Coefficients:
       beta0           beta1           beta2
poly.1 3.2017 (0.0182)
poly.2 1.9077 (0.0217) 0.4004 (0.0069)
poly.3 1.6103 (0.0279) 0.6756 (0.0209) -0.0335 (0.0024)
sing.3 0.9280 (0.0287) 1.3860 (0.0203)  1.0000 (0.0000)

Model Fitting:
       rss     df pfit   aic
poly.1 3690.22 12 0.0000 3666.22
poly.2  220.16 11 0.0000  198.16
poly.3   37.23 10 0.0001   17.23
```

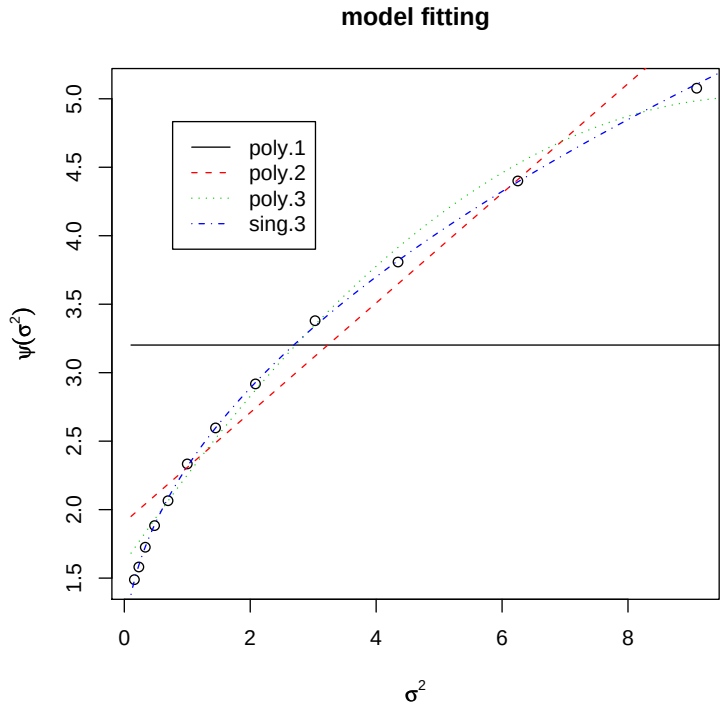

FIGURE 1. Model Fitting

```
sing.3    1.95 10 0.9967  -18.05

Best Model:  sing.3
```

The results of (i) are the BPs for the 13 scales shown at first. Let α_{σ^2} denote the BP at scale σ^2. Each BP is calculated from 10,000 bootstrap samples of size n' as the frequency of observing $f(y^*) = 1$. In `scaleboot`, n' is `round(n/sa[i])`, for $i = 1, \ldots, 13$. Then, the scale is recalculated as $\sigma^2 = n/n'$ for taking account of the discreteness. The actual labor of `scaleboot` function is in fact only step (i).

The step (ii) is performed by `sbfit` function internally for fitting parametric models to observed α_{σ^2}'s. By default, four models are considered as candidates; `poly.1`, `poly.2`, `poly.3`, and `sing.3`. Each of these models are denoted as $\psi(\sigma^2|\beta)$. Let $z_{\sigma^2} = \Phi^{-1}(1 - \alpha_{\sigma^2})$ be the bootstrap z-value at scale σ^2, where $\Phi^{-1}(p)$=`qnorm(p)`. The coefficient vector β is estimated by fitting $\sigma z_{\sigma^2} = \psi(\sigma^2|\beta)$. Let $\hat{\beta}$ denote the estimated value; the detail of fitting is explained later. We may choose the model which minimizes AIC value. The fitted curves are shown (Fig. 1) by plotting $\psi(\sigma^2|\hat{\beta})$ as

```
> plot(X.sb, legend = "topleft")
```

The same plot but in other variables can be shown (Fig. 2) by, for example,

```
> plot(X.sb, xval = "sigma", log = "x", yval = "pvalue", legend = "topleft")
```

`poly.k` model is specified as a polynomial of σ^2; $\psi(\sigma^2|\beta) = \sum_{j=0}^{k-1} \beta_j \sigma^{2j}$ for $k \geq 1$. `sing.k` model is specified as $\psi(\sigma^2|\beta) = \beta_0 + \sum_{j=1}^{k-2} \beta_j \sigma^{2j}(1 + a)/(1 + a\sigma)$ for $k \geq 3$, where $a = \beta_{k-1}/(1 - \beta_{k-1})$ so that $0 \leq a \leq \infty$ for $0 \leq \beta_{k-1} \leq 1$. The number `k` for each model denotes the number of coefficients in β.

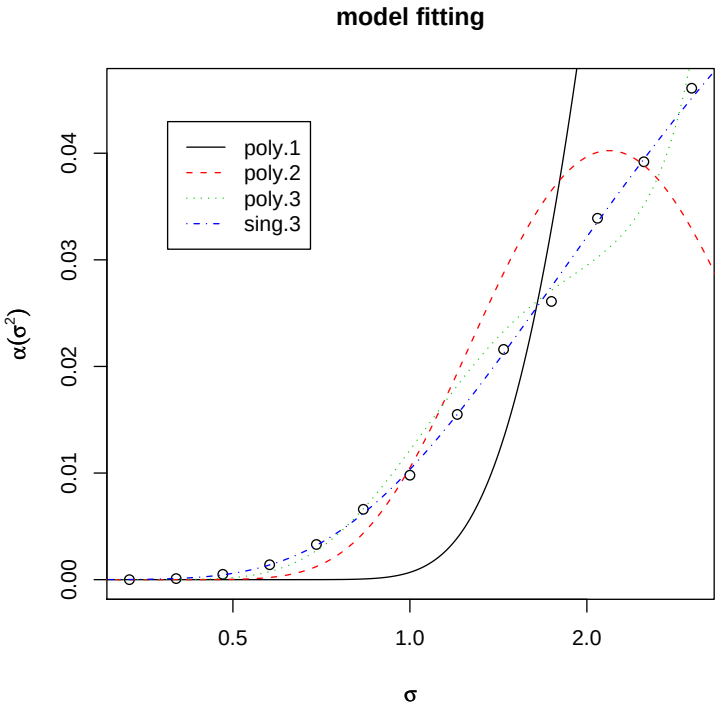

FIGURE 2. Model Fitting ($x = \log\sigma$, $y = \alpha_{\sigma^2}$)

The detail of model fitting is as follows. Let B_i and C_i be the number of replicates and the observed number of times that $f(y^*) = 1$, respectively, for the bootstrap resampling of scale σ_i^2, $i = 1, \ldots, S$. Since each C_i is distributed as binomial, the log-likelihood is

$$\ell(\beta) = \sum_{i=1}^{S} \Big\{ C_i \log \Phi(-\psi(\sigma_i^2|\beta)/\sigma_i) + (B_i - C_i) \log \Phi(\psi(\sigma_i^2|\beta)/\sigma_i) \Big\},$$

where $\Phi(q)$=`pnorm(q)`. The estimate $\hat{\beta}$ is obtained by maximizing $\ell(\beta)$ numerically. The goodness of fit is measured by the difference of AIC values between the specified model and an unconstrained binomial model;

$$\mathrm{AIC} = (-2\ell(\hat{\beta}) + 2k) - (-2\hat{\ell} + 2S),$$

where $\hat{\ell} = \sum_{i=1}^{S} (C_i \log(C_i/B_i) + (B_i - C_i) \log(1 - C_i/B_i))$.

The step (iii) is performed by the the `summary` method as already mentioned. The first line shows the "raw" BP α_1 (the BP obtained from the ordinary bootstrap resampling). The main results are the corrected p-values follow next. For each model, we calculate q_k, $k = 1, 2, 3$, by

$$q_k = \sum_{j=0}^{k-1} \frac{(-1 - \sigma_0^2)^j}{j!} \frac{\partial^j \psi(\sigma^2|\hat{\beta})}{\partial(\sigma^2)^j}\Big|_{\sigma_0^2}.$$

Then the corrected p-values are calculated by $p_k = 1 - \Phi(q_k)$. By default $\sigma_0^2 = 1$. The calculation of q_k is interpreted as extrapolation of σz_{σ^2} to $\sigma^2 = -1$ by using the first k terms of the Taylor series. According to the theory of Shimodaira (2006), the

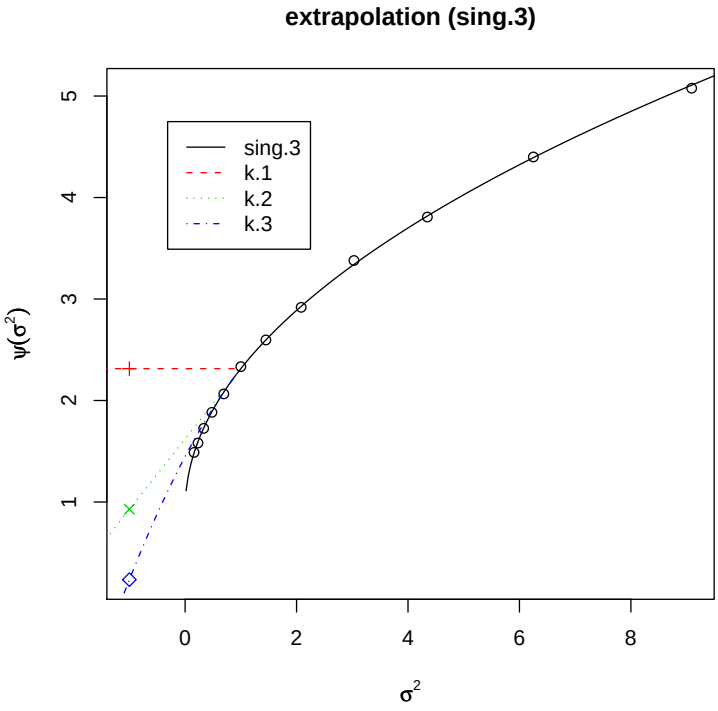

FIGURE 3. Extrapolation

unbiased p-value is, if exists, obtained by taking the limit $k \to \infty$. The extrapolated curves are shown (Fig. 3) by

```
> plot(summary(X.sb), legend = "topleft")
```

4. Hierarchical Clustering

4.1. **Pvclust Package.** `scaleboot` package includes an interface to `pvclust` package (Suzuki and Shimodaira 2006) of R for bootstrapping hierarchical clustering. We use `pvclust` to calculate multiscale BPs for clusters, from which we calculate an improved version of AU p-values using `scaleboot`. See `help(lung73)` for further details of the following example.

4.2. **Using Pvclust.** We work on `lung` dataset (Garber et al. 2001) included in `pvclust`. It is a DNA microarray dataset of 73 lung tissues (arrays) with 916 observations of genes. To draw dendrograms in terms of the arrays, we resample genes in our analysis; this may be interpreted as assessing the uncertainty due to variability of genes. The function `pvclust` first obtains a dendrogram by a hierarchical clustering method, and then calculates the multiscale BPs for each cluster of the dendrogram.

```
> library(pvclust)
> data(lung)
> sa <- 9^seq(-1, 1, length = 13)
> nb <- 10000
> lung73.pvclust <- pvclust(lung, r = 1/sa, nboot = nb)
```

The above code may take a day, so it would be a good idea to run with nb=1000 so that 10 times faster. However, nb=1000 should be just for checking the program, and nb=10,000 (at least) is recommended for publishing the result.

4.3. **Model Fitting.** We next apply `sbfit` function of `scaleboot` to the multiscale BPs. For each cluster of the dendrogram, parametric models are fitted to the BPs.

```
> library(scaleboot)
> lung73.sb <- sbfit(lung73.pvclust)
```

4.4. **Lung73 Dataset.** The results of the previous two sections (`lung73.pvclust` and `lung73.sb`) are in fact stored in `lung73` dataset of `scaleboot`. For users who want to try the examples, just type as follows.

```
> library(scaleboot)
> data(lung73)
```

We have used a cluster computer of 40 cpus for parallel computing using `snow` package. The following code may take only an hour.

```
> library(snow)
> cl <- makeCluster(40)
> library(pvclust)
> data(lung)
> sa <- 9^seq(-1, 1, length = 13)
> nb <- 10000
> lung73.pvclust <- parPvclust(cl, lung, r = 1/sa, nboot = nb)
> library(scaleboot)
> lung73.sb <- sbfit(lung73.pvclust, cluster = cl)
```

4.5. P**-value Calculation.** To calculate AU p-values (p_3) from `lung73.sb` and write them back to `lung73.pvclust`, we do

```
> lung73.k3 <- sbpvclust(lung73.pvclust, lung73.sb)
```

To see the results, we simply plot the dendrogram (Fig. 4) by

```
> library(pvclust)
> plot(lung73.k3, cex = 0.5, cex.pv = 0.7)
> pvrect(lung73.k3)
```

To calculate p_2 instead of p_3, specify `k=2`,

```
> lung73.k2 <- sbpvclust(lung73.pvclust, lung73.sb, k = 2)
```

4.6. **Diagnostics of Fitting.** The fitted curves are drawn by the plot method. For node 67, say, a plot with legend is obtained (Fig. 5) by

```
> plot(lung73.sb[[67]], legend = "topleft")
```

All the calculated p-values for node 67 are given by

```
> summary(lung73.sb[[67]])
```

```
Raw Bootstrap Probability:  3.63 (0.19)

Corrected P-values (percent):
       k.1          k.2          k.3          aic
poly.1 18.41 (0.10) 18.41 (0.10) 18.41 (0.10) 52878.29
poly.2  5.46 (0.09) 83.51 (0.32) 83.51 (0.32)  1356.63
poly.3  3.95 (0.08) 86.05 (0.29) 92.56 (0.28)   464.71
sing.3  3.31 (0.07) 77.02 (0.47) 95.10 (0.17)    25.11

Best Model:  sing.3
```

The extrapolation using the best model is shown (Fig. 6) by

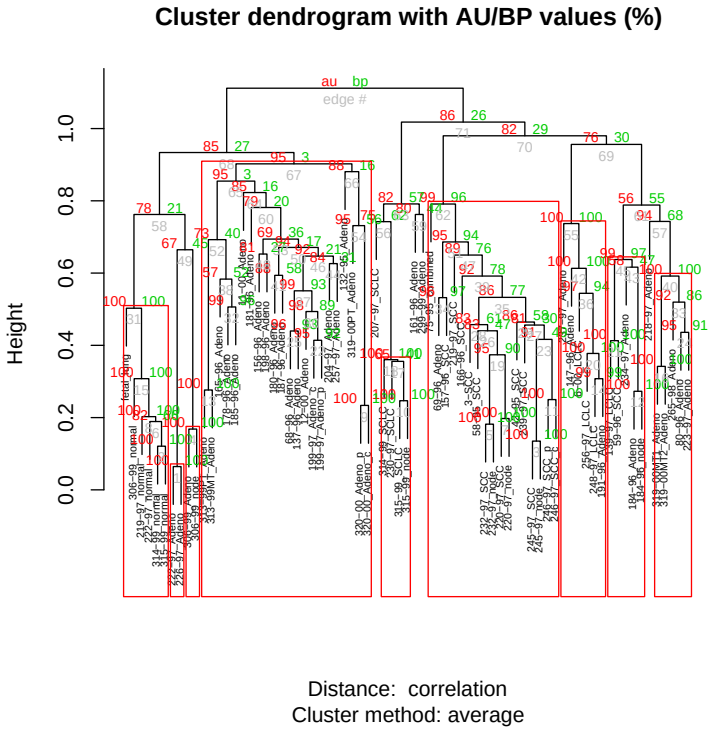

FIGURE 4. Dendrogram of lung73 dataset

```
> plot(summary(lung73.sb[[67]]), legend = "topleft")
```

For a set of nodes, p-values are given by

```
> summary(lung73.sb[c(62, 67, 69, 71)])
Corrected P-values (percent):
   raw          k.1          k.2          k.3          model  aic
62 95.68 (0.20) 95.92 (0.10) 98.64 (0.10) 98.61 (0.12) poly.3 -12.01
67  3.63 (0.19)  3.31 (0.07) 77.02 (0.47) 95.10 (0.17) sing.3  25.11
69 29.49 (0.46) 29.65 (0.17) 75.37 (0.22) 75.83 (0.34) poly.3 -14.09
71 25.20 (0.43) 25.95 (0.17) 84.44 (0.18) 85.91 (0.27) poly.3  11.49
```

Also plots are shown (Fig. 7) by

```
> plot(lung73.sb[c(62, 67, 69, 71)])
```

5. PHYLOGENETIC INFERENCE

5.1. **CONSEL Software.** `scaleboot` has a front end for phylogenetic inference, and it may eventually replace `CONSEL` software (Shimodaira and Hasegawa 2001) for testing phylogenetic trees. Currently, `scaleboot` does not have a method for file conversion of several phylogenetic software, and so we must use `CONSEL` for this purpose before applying `scaleboot` to calculate an improved version of AU p-values for trees and edges. See `help(mam15)` for further details of the following example.

5.2. **Mammal Dataset.** We work on 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 dataset was originally used in Shimodaira and Hasegawa (1999).

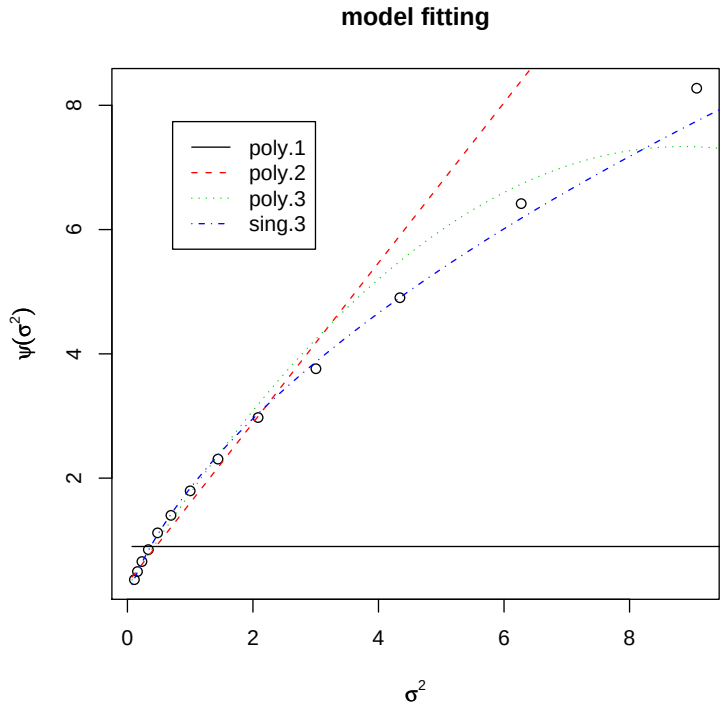

FIGURE 5. Model fitting for node 67

For Unix users, download `mam15-files.tgz`, and for Windows users download `mam15-files.zip`. The details of dataset files are as follows. `mam15.aa`: amino acid sequences ($n = 3414$) of mtDNA for the six mammals. `mam15.ass`: association vectors for edges and trees. `mam15.lnf`: site-wise log-likelihood values (output from PAML). `mam15.log`: detailed information for the associations. `mam15.mt`: site-wise log-likelihood values (output from seqmt). `mam15.tpl`: 15 tree topologies.

5.3. **Likelihood Calculation of Trees.** The main body of the dataset is the amino acid sequences (`mam15.aa`). We consider $m = 15$ tree topologies of the six mammals (`mam15.tpl`);

```
((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
```

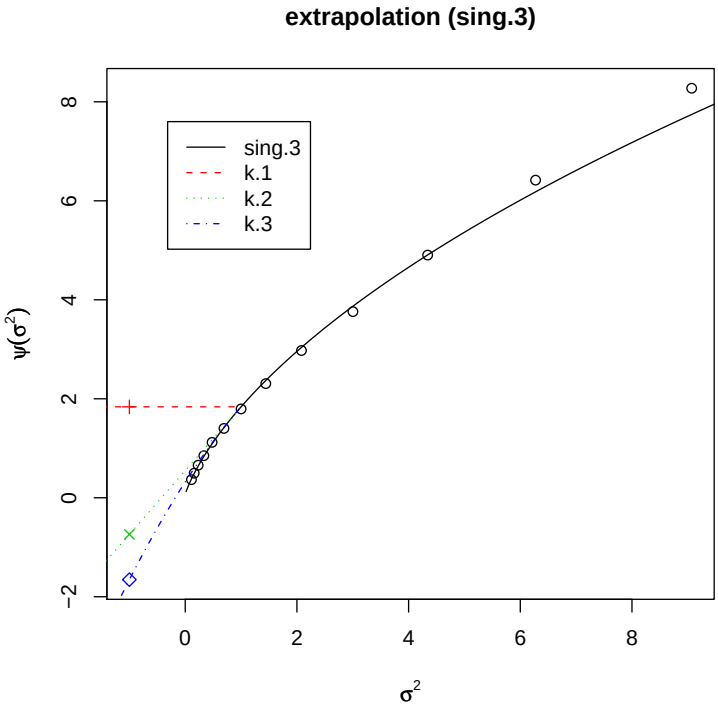

FIGURE 6. Extrapolation for node 67

The maximum likelihood estimates for these trees are calculated by PAML (Yang 1997). Let x_{ij} be the site-wise log-likelihood for sites $i = 1, \ldots, n$, and trees $j = 1, \ldots, m$. The log-likelihood of tree-j is $\sum_{i=1}^{n} x_{ij}$. The large n justifies the central limit theorem for $y = \bar{x}$, and allows us to resample x_{ij} directly without recalculation of the maximum likelihood estimates. The matrix $X = (x_{ij})$ is produced by PAML and stored in `mam15.lnf`. It is converted by CONSEL to a simpler format and stored in `mam15.mt`. The command is

```
seqmt --paml mam15.lnf
```

5.4. ***P*-value Calculation for Trees.** The AU p-values for trees are calculated simply by

```
> library(scaleboot)
> mam15.mt <- read.mt("mam15.mt")
> mam15.trees <- relltest(mam15.mt)
> summary(mam15.trees)
```

The `relltest` function above may take a half hour. The next section can be skipped if only tree selection is of interest.

5.5. ***P*-value Calculation for Clusters.** We can also calculate AU p-values for clusters (edges) of trees. We have to know, for each cluster, in which of the 15 trees it is included. The file `mam15.ass` has this information, which was generated using CONSEL by the command

```
treeass --outgroup 6 mam15.tpl > mam15.log
```

It also produces `mam15.log` for human readable information. A part of `mam15.log` is as follows.

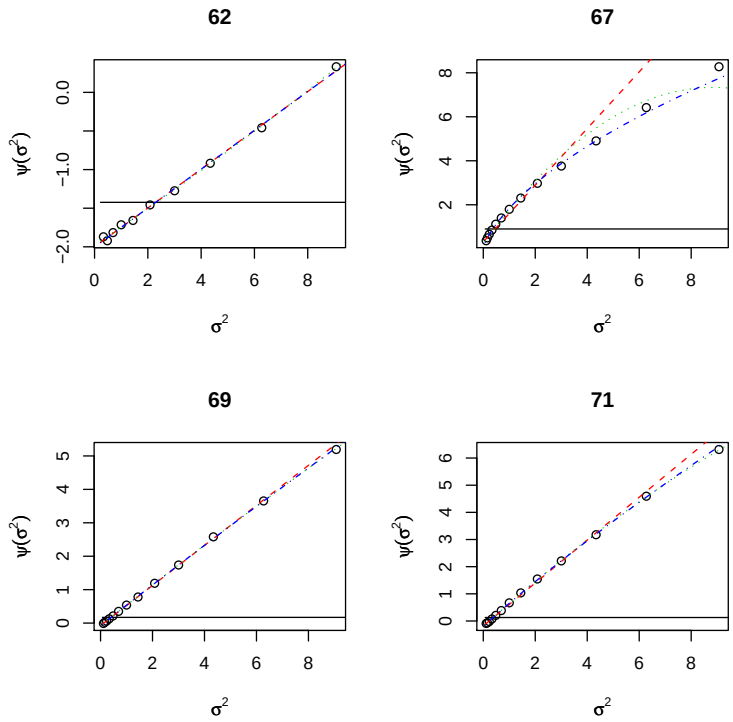

FIGURE 7. Model fitting for a set of nodes

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

# base edges: 10
10 6

    123456
  1 +++---  ; (0.2000)
  2 ++++--  ; (0.2000)
  3 +--+--  ; (0.2000)
  4 -+++--  ; (0.2000)
  5 ---++-  ; (0.2000)
  6 +--++-  ; (0.2000)
  7 -++++-  ; (0.2000)
  8 +++-+-  ; (0.2000)
  9 +---+-  ; (0.2000)
 10 -++-+-  ; (0.2000)
```

The above defines clusters (edges) named e1,...e10. For example, e1 = +++-- = (Homsa, Phovi, Bosta).

The AU p-values for clusters as well as trees are calculated simply by

```
> library(scaleboot)
> mam15.mt <- read.mt("mam15.mt")
> mam15.ass <- read.ass("mam15.ass")
> mam15.relltest <- relltest(mam15.mt, ass = mam15.ass)
> summary(mam15.relltest)
```

5.6. **Mam15 Dataset.** The results of the previous sections (`mam15.mt`, `mam15.ass`, and `mam15.relltest`) are in fact stored in `mam15` dataset of `scaleboot`. For users who want to try the examples, just type as follows.

```
> library(scaleboot)
> data(mam15)
```

The results for trees are extracted by

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

We have used a cluster computer of 40 cpus for parallel computing using `snow` package. The following code may take only 10 minutes, although we have used the number of resamples ten times larger than the default value.

```
> library(snow)
> cl <- makeCluster(40)
> library(scaleboot)
> mam15.mt <- read.mt("mam15.mt")
> mam15.ass <- read.ass("mam15.ass")
> mam15.relltest <- relltest(mam15.mt, nb = 1e+05, ass = mam15.ass)
```

5.7. **Interpreting the Results.** First we sort the results in increasing order of log-likelihood difference,

```
> stat <- attr(mam15.trees, "stat")
> o <- order(stat)
> mam15.trees <- mam15.trees[o]
> 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
```

Next we look at p-values. We confirm that p_1 (the second column) is almost the same as the raw BP (the first column); this should be so if the model fitting is good. Only two trees, i.e., t1 and t3, have $p_1 > 0.05$. It is known that the bias of

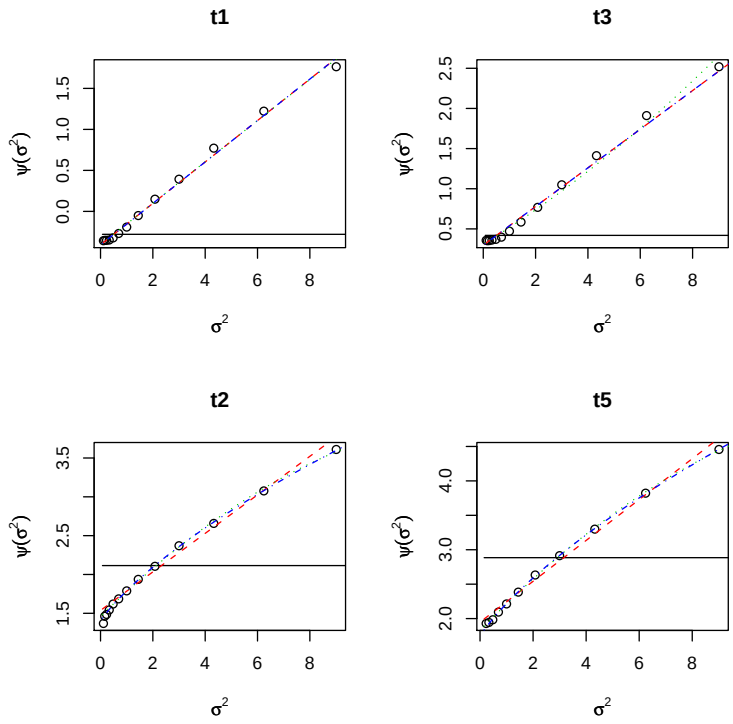

FIGURE 8. Model fitting for the top four trees

p_1 is large so that often leads to false positives for tree selection. p_2 improves p_1 by correcting the bias. Six trees, i.e., t1, t3, t2, t5, t6, and t4, have $p_2 > 0.05$. p_3 improves p_2 even more, although the trees of $p_3 > 0.05$ are the same six trees in this example.

Finally we examine model fitting. According to AIC values, the fitting is good overall except for the top two trees; however note that the AIC values should be about 10 times smaller if the default nb=10,000 were used. The fitting curves for the top four trees are shown (Fig. 8) by

```
> plot(mam15.trees[1:4])
```

According to the plots, the fitting is rather good even to the top two trees.

REFERENCES

[1] Felsenstein, J. (1985). Confidence limits on phylogenies: an approach using the bootstrap. *Evolution* **39** 783–791.

[2] 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/adeno/`).

[3] Shimodaira, H. (2002). An approximately unbiased test of phylogenetic tree selection. *Syst. Biol.* **51** 492–508.

[4] Shimodaira, H. (2006) Approximately Unbiased Tests for Singular Surfaces via Multiscale Bootstrap Resampling. Research Report B-430, Dept. Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo.

[5] 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/`).

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

[7] 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`).

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan

E-mail address: `shimo@is.titech.ac.jp`