# Package 'Iboot'

February 19, 2015

**Version** 0.1-1

**Date** 2013-02-15

**Title** Iboot: iterated bootstrap tests and confidence sets.

**Author** Nicola Lunardon <`nicola.lunardon@econ.units.it`>

**Maintainer** Nicola Lunardon <`nicola.lunardon@econ.units.it`>

**Description** The package implements a general algorithm to obtain iterated bootstrap tests and confidence sets for a p-dimensional parameter based on the unstudentized version of the Rao statistic.

**License** GPL-2

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2013-02-15 14:20:27

## R topics documented:

## `Iboot-package` *Iboot: iterated bootstrap tests and confidence sets*

### Description

The package provides a computationally efficient and general algorithm to obtain iterated bootstrap tests and confidence sets based on the unstudentised version of the Rao statistic for a $p$-dimensional parameter. The outer and inner level of resampling required to obtain respectively the simple and the re-calibrated bootstrap critical values (at the null hypothesys) are performed in a weighted fashion. The particular choice of the resampling weights allows to obtain accurate re-calibrated critical values with one level of bootstrap iteration only (Lee and Young, 2003).

The algorithm is particularly efficient as it combines a deterministic stopping rule (Nankervis, 2005) and a computationally convenient statistic to bootstrap on (Lunardon, 2013).

## Details

Function `Iboot` is merely an R wrapper to call a set of foreign functions all written in C language so that computational efficiency is increased. Some C routines are borrowed from R sources: numerical optimisation and sorting relies on `lbfgsb` and `revsort` located in "/src/main/optim.c" and "/src/main/sort.c", respectively. The function `ProbSampleReplace` for sampling with unequal probabilities has been slightly modified to cut down the number of unnecessary operations for bootstrap resamplings.

## Author(s)

Nicola Lunardon <nicola.lunardon@econ.units.it>.

Maintainer: Nicola Lunardon <nicola.lunardon@econ.units.it>.

## References

Lee, S., Young, A. (2003). Prepivoting by weighted bootstrap iteration. *Biometrika*, **90**, 393–410.

Lunardon, N. (2013). Prepivoting composite score statistics by weighted bootstrap iteration. E-print: arXiv/1301.7026.

Nankervis, J. (2005). Computational algorithms for double bootstrap confidence intervals. *Computational statistics & data analysis*, **45**, 461–475.

## See Also

`one.boot`, `boot`, `stats`.

## Examples

```
####Example 1: mean of a normal with known scale
n <- 20
mu <- 1

set.seed(1)
##contributions obtained from the score function
gr <- rnorm(n, mu) - mu

OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05, 0.01), seed=1)

##critical values for testing H0: mu=1, H1: mu!=1
OBJ.Ib
summary(OBJ.Ib)

####Example 2: variance of a normal with known location
n <- 20
mu <- 1
sig2 <- 1
```

```
set.seed(1)
##contributions obtained from the score function
gr <- ( rnorm(n, mu, sqrt(sig2)) - mu )^2/sig2 - 1

OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05, 0.01), seed=3)

##critical values for testing H0: sig2=1, H1: sig2!=1
OBJ.Ib
summary(OBJ.Ib)
```

## Iboot — *Iterated bootstrap tests and confidence sets for a $p$-dimensional parameter*

### Description

This function provides both simple and re-calibrated bootstrap critical values for tests and confidence sets for a $p$-dimensional parameter, obtained by a single weighted bootstrap iteration. The considered statistic is the unstudentised version of the Rao statistic as proposed in Lunardon (2013). Details about the implemented algorithm can be found in Nankervis (2005) and Lunardon (2013).

### Usage

```
Iboot(gradient, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05), control.optim=list(), seed)
```

### Arguments

| | |
|---|---|
| `gradient` | An object of class `data.matrix`, or coercible to that class, of dimension `n x p`, where `n` is the sample size and `p` is the dimension of the parameter, containing the contributions of an estimating function for the parameter evaluated at the desired value. See “Details”. |
| `B` | An integer indicating the number of outer level bootstrap replications. |
| `M` | An integer indicating the number of inner level bootstrap replications. If equal to 0 then single bootstrap is performed. |
| `kB` | The proportion of convex hull condition failures in the outer level that the user allows before stopping the algorithm. See “Details”. |
| `alpha` | A vector specifying the desired nominal level(s) of test and confidence set. |
| `control.optim` | A list of optional control parameters as passed to the optimisation routine `optim` for `method="L-BFGS-B"`. See “Details”. |
| `seed` | A single value, interpreted as an integer, recommended to specify seeds. |

## Details

`Iboot` performs a single weighted bootstrap iteration in order to provide re-calibrated critical values for tests and confidence sets based on the unstudentised version of the Rao statistic for a parameter $\theta \in R^p$, $p \geq 1$.

Denoted with $y = (y_1, \ldots, y_n)$ an i.i.d. sample of size $n$, where $y_i \in R^d$, $d \geq 1$, and with $U(\theta) = \sum_{i=1}^n u(\theta; y_i)$ an (unbiased) estimating function for $\theta$, the unstudentised version of the Rao statistic is defined as

$$W_{us}(\theta) = n^{-1} U(\theta)^\top U(\theta).$$

The function `Iboot` implements the algorithm outlined in Lunardon (2013) that differs from the Nankervis's one for two sided confidence intervals because bootstrap is carried out in a weighted fashion. In the outer level bootstrap versions of $W_{us}(\theta)$, $W_{us}^b(\theta)$, $b = 1, \ldots, B$, are obtained by resampling according to the $n$-dimensional vector of weights $w(\theta) = (w_1(\theta), \ldots, w_n(\theta))$ associated to each contribution $u(\theta; y_i)$. The functional form of $w_i(\theta)$ is provided by Owen's empirical likelihood formulation, that is

$$w_i(\theta) = n^{-1}(1 + \lambda(\theta)^\top u(\theta; y_i))^{-1},$$

where $\lambda(\theta) \in R^p$ is the Lagrange multiplier (see Owen, 1990). Instead, in the inner level, bootstrap versions of $W_{us}^b(\theta)$ are obtained by resampling according to $w^b(\theta)$, computed as before by using the outer level contributions $u(\theta; y_i^b)$.

As weighted bootstrap might entail computational difficulties, i.e. the convex hull condition (see Owen, 1990) might not be satisfied in all `B` bootstrap replications, some precautions have been taken. In particular, the algorithm checks if the convex hull condition is satisfied for both the observed contributions $u(\theta; y_i)$ and for each of the `B` bootstrap resamplings in the outer level, i.e. $u(\theta; y_i^b)$.

The algorithm stops immediately if the convex hull condition fails for $u(\theta; y_i)$ whereas concludes after reaching `B x kB` convex hull condition failures in the outer level. In particular, in the former situation the observed value of the statistic is returned only, whereas in the latter both the observed value and the bootstrap distribution (based on a smaller number of resamplings than `B`) of the statistic are supplied.

Numerical optimisation of the objective function that supplies $\lambda(\theta)$ relies on the foreign function `lbfgsb` called from R by `optim` with `method="L-BFGS-B"`. However, it is not possible to set bounds on the variables by specifying `lower` and `upper`, instead the optional parameters that can be set in `control.optim` are

`trace` Non-negative integer. If positive, tracing information on the progress of the optimisation is produced. Higher values may produce more tracing information (up to six levels of tracing). Defaults to 0.

`maxit` The maximum number of iterations. Defaults to 2e4.

`pgtol` It is a tolerance on the projected gradient in the current search direction. Defaults to `sqrt(.Machine$double.eps)`.

`REPORT` The frequency of reports if `trace` is positive. Defaults to every 10 iterations.

`lmm` An integer giving the number of BFGS updates retained. Defaults to 5.

`factr` Controls the convergence. Convergence occurs when the reduction in the objective is within this factor of the machine tolerance. Default is 1e7, that is a tolerance of about 1e-8.

## Value

A list of class “Iboot” containing:

| | |
|---|---|
| `Call` | The matched call. |
| `oss` | The observed value of the statistic. |
| `boot` | The bootstrap distribution of the statistic (sorted into descending order). |
| `map` | The re-calibrated `1-alpha` nominal levels. |
| `boot.quant` | The `1-alpha` level bootstrap quantile(s). |
| `recalib.quant` | The re-calibrated `1-alpha` level bootstrap quantile(s). |
| `fails.outer` | Actual proportion of failures in the outer level. |
| `failure` | An error code indicating wheter the algorithm has succeeded: |

- `0` indicates that the algorithm have reached the end successfully (see `fails.outer` to check the actual proportion of failures).
- `1` convex hull condition not satisfied by using the original contributions $u(\theta; y_i)$.
- `2` the actual proportion of convex hull condition failures in the outer level has exceeded `kB`.

## Note

S3 print and summary methods are associated to objects of class “Iboot”.

## References

Lunardon, N. (2013). Prepivoting composite score statistics by weighted bootstrap iteration. E-print: arXiv/1301.7026.

Nankervis, J. (2005). Computational algorithms for double bootstrap confidence intervals. *Computational statistics & data analysis*, **45**, 461–475.

Owen, A. (1990). Empirical likelihood ratio confidence regions. *The Annals of Statistics*, **18**, 90–120.

## See Also

`one.boot`, `boot`, `optim`

## Examples

```
####Example 1: mean of a normal with known scale
n <- 20
mu <- 1

set.seed(1)
##contributions obtained from the score function
gr <- rnorm(n, mu) - mu

OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05, 0.01), seed=1)
```

```
##critical values for testing H0: mu=1, H1: mu!=1
OBJ.Ib
summary(OBJ.Ib)

#####exceeded number of convex hull condition failures in the outer level (kB=0)
OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0, alpha=c(0.1, 0.05, 0.01), seed=1)
OBJ.Ib
summary(OBJ.Ib)

## Not run:
####Example 2: example 1 of Lunardon (2013)
n <- 20
q <- 10

##parameter
mu <- 0
sig2 <- 1
rho <- 0.5
theta <- c(mu, sig2, rho)

##function to compute the pairwise score contributions
pscore_theta <- function(theta, data)
{
n <- nrow(data)
q <- ncol(data)
mu <- theta[1]
sig2 <- theta[2]
rho <- theta[3]
A <- matrix(rho, q, q)
diag(A) <- -(q-1)
A <- A/((1-rho^2)*sig2^2)
B <- matrix(-(1+rho^2), q, q)
diag(B) <- 2*rho*(q-1)
B <- B/(sig2*(1-rho^2)^2)
x_dot <- apply(data, 1, sum)
p_mu <- ((q-1)/(sig2*(1+rho)))*(x_dot-q*mu)
p_sig2 <- -0.5*apply(((data-mu)
p_rho <- q*(q-1)*rho*0.5/(1-rho^2)-0.5*apply(((data-mu)
RES <- cbind(p_mu, p_sig2, p_rho)
colnames(RES) <- c("mu", "sig2", "rho")
RES
}


###data simulation

##correlation matrix
S <- matrix(rho*sig2, q, q)
diag(S) <- sig2
##cholesky
cholS <- chol(S)

##generation from multivariate normal
```

```
set.seed(3)
Y <- matrix(rnorm(n*q), n, q)

##compute pairwise score conributions
gr <- pscore_theta(theta, Y)

OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05, 0.01), seed=3)

##critical values for testing H0: theta=(0, 1, 0.5), H1: theta!=(0, 1, 0.5)
OBJ.Ib
summary(OBJ.Ib)

##sampe size too small: convex hull failure at the beginning
n <- 10
set.seed(3)
Y <- matrix(rnorm(n*q), n, q)

##compute pairwise score conributions
gr <- pscore_theta(theta, Y)

OBJ.Ib <- Iboot(gradient=gr, B=500, M=500, kB=0.01, alpha=c(0.1, 0.05, 0.01), seed=3)
OBJ.Ib
summary(OBJ.Ib)

## End(Not run)
```

# Index