Package 'boot'

April 21, 2017

Priority recommended

Version 1.3-19

Date 2017-02-11

Maintainer Brian Ripley <ripley@stats.ox.ac.uk>

Note Maintainers are not available to give advice on using a package
they did not author.

Description Functions and datasets for bootstrapping from the
book ``Bootstrap Methods and Their Application'' by A. C. Davison and
D. V. Hinkley (1997, CUP), originally written by Angelo Canty for S.

Title Bootstrap Functions (Originally by Angelo Canty for S)

Depends R (>= 3.0.0), graphics, stats

Suggests MASS, survival

LazyData yes

ByteCompile yes

License Unlimited

NeedsCompilation no

Author Angelo Canty [aut],
Brian Ripley [aut, trl, cre] (author of parallel support)

Repository CRAN

Date/Publication 2017-04-21 13:33:28 UTC

R topics documented:

`abc.ci` *Nonparametric ABC Confidence Intervals*

Description

Calculate equi-tailed two-sided nonparametric approximate bootstrap confidence intervals for a parameter, given a set of data and an estimator of the parameter, using numerical differentiation.

Usage

```
abc.ci(data, statistic, index=1, strata=rep(1, n), conf=0.95,
       eps=0.001/n, ...)
```

Arguments

data	A data set expressed as a vector, matrix or data frame.
statistic	A function which returns the statistic of interest. The function must take at least 2 arguments; the first argument should be the data and the second a vector of weights. The weights passed to `statistic` will be normalized to sum to 1 within each stratum. Any other arguments should be passed to `abc.ci` as part of the `...{}` argument.
index	If `statistic` returns a vector of length greater than 1, then this indicates the position of the variable of interest within that vector.
strata	A factor or numerical vector indicating to which sample each observation belongs in multiple sample problems. The default is the one-sample case.
conf	A scalar or vector containing the confidence level(s) of the required interval(s).
eps	The value of epsilon to be used for the numerical differentiation.
...	Any other arguments for `statistic`. These will be passed unchanged to `statistic` each time it is called within `abc.ci`.

Details

This function is based on the function `abcnon` written by R. Tibshirani. A listing of the original function is available in DiCiccio and Efron (1996). The function uses numerical differentiation for the first and second derivatives of the statistic and then uses these values to approximate the bootstrap BCa intervals. The total number of evaluations of the statistic is `2*n+2+2*length(conf)` where `n` is the number of data points (plus calculation of the original value of the statistic). The function works for the multiple sample case without the need to rewrite the statistic in an artificial form since the stratified normalization is done internally by the function.

Value

A `length(conf)` by 3 matrix where each row contains the confidence level followed by the lower and upper end-points of the ABC interval at that level.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*, Chapter 5. Cambridge University Press.

DiCiccio, T. J. and Efron B. (1992) More accurate confidence intervals in exponential families. *Biometrika*, **79**, 231–245.

DiCiccio, T. J. and Efron B. (1996) Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.

See Also

`boot.ci`

Examples

```
# 90% and 95% confidence intervals for the correlation
# coefficient between the columns of the bigcity data

abc.ci(bigcity, corr, conf=c(0.90,0.95))

# A 95% confidence interval for the difference between the means of
# the last two samples in gravity
mean.diff <- function(y, w)
{    gp1 <- 1:table(as.numeric(y$series))[1]
     sum(y[gp1, 1] * w[gp1]) - sum(y[-gp1, 1] * w[-gp1])
}
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
abc.ci(grav1, mean.diff, strata = grav1$series)
```

---

acme *Monthly Excess Returns*

Description

The acme data frame has 60 rows and 3 columns.

The excess return for the Acme Cleveland Corporation are recorded along with those for all stocks listed on the New York and American Stock Exchanges were recorded over a five year period. These excess returns are relative to the return on a risk-less investment such a U.S. Treasury bills.

Usage

```
acme
```

Format

This data frame contains the following columns:

`month` A character string representing the month of the observation.

`market` The excess return of the market as a whole.

`acme` The excess return for the Acme Cleveland Corporation.

Source

The data were obtained from

Simonoff, J.S. and Tsai, C.-L. (1994) Use of modified profile likelihood for improved tests of constancy of variance in regression. *Applied Statistics*, **43**, 353–370.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

aids *Delay in AIDS Reporting in England and Wales*

Description

The aids data frame has 570 rows and 6 columns.

Although all cases of AIDS in England and Wales must be reported to the Communicable Disease Surveillance Centre, there is often a considerable delay between the time of diagnosis and the time that it is reported. In estimating the prevalence of AIDS, account must be taken of the unknown number of cases which have been diagnosed but not reported. The data set here records the reported cases of AIDS diagnosed from July 1983 and until the end of 1992. The data are cross-classified by the date of diagnosis and the time delay in the reporting of the cases.

Usage

```
aids
```

Format

This data frame contains the following columns:

`year` The year of the diagnosis.

`quarter` The quarter of the year in which diagnosis was made.

`delay` The time delay (in months) between diagnosis and reporting. 0 means that the case was reported within one month. Longer delays are grouped in 3 month intervals and the value of `delay` is the midpoint of the interval (therefore a value of 2 indicates that reporting was delayed for between 1 and 3 months).

`dud` An indicator of censoring. These are categories for which full information is not yet available and the number recorded is a lower bound only.

`time` The time interval of the diagnosis. That is the number of quarters from July 1983 until the end of the quarter in which these cases were diagnosed.

`y` The number of AIDS cases reported.

Source

The data were obtained from

De Angelis, D. and Gilks, W.R. (1994) Estimating acquired immune deficiency syndrome accounting for reporting delay. *Journal of the Royal Statistical Society, A*, **157**, 31–40.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

aircondit *Failures of Air-conditioning Equipment*

Description

Proschan (1963) reported on the times between failures of the air-conditioning equipment in 10 Boeing 720 aircraft. The `aircondit` data frame contains the intervals for the ninth aircraft while `aircondit7` contains those for the seventh aircraft.

Both data frames have just one column. Note that the data have been sorted into increasing order.

Usage

```
aircondit
```

Format

The data frames contain the following column:

`hours` The time interval in hours between successive failures of the air-conditioning equipment

Source

The data were taken from

Cox, D.R. and Snell, E.J. (1981) *Applied Statistics: Principles and Examples*. Chapman and Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Proschan, F. (1963) Theoretical explanation of observed decreasing failure rate. *Technometrics*, **5**, 375-383.

amis *Car Speeding and Warning Signs*

Description

The `amis` data frame has 8437 rows and 4 columns.

In a study into the effect that warning signs have on speeding patterns, Cambridgeshire County Council considered 14 pairs of locations. The locations were paired to account for factors such as traffic volume and type of road. One site in each pair had a sign erected warning of the dangers of speeding and asking drivers to slow down. No action was taken at the second site. Three sets of measurements were taken at each site. Each set of measurements was nominally of the speeds of 100 cars but not all sites have exactly 100 measurements. These speed measurements were taken before the erection of the sign, shortly after the erection of the sign, and again after the sign had been in place for some time.

Usage

```
amis
```

Format

This data frame contains the following columns:

`speed` Speeds of cars (in miles per hour).

`period` A numeric column indicating the time that the reading was taken. A value of 1 indicates a reading taken before the sign was erected, a 2 indicates a reading taken shortly after erection of the sign and a 3 indicates a reading taken after the sign had been in place for some time.

`warning` A numeric column indicating whether the location of the reading was chosen to have a warning sign erected. A value of 1 indicates presence of a sign and a value of 2 indicates that no sign was erected.

`pair` A numeric column giving the pair number at which the reading was taken. Pairs were numbered from 1 to 14.

Source

The data were kindly made available by Mr. Graham Amis, Cambridgeshire County Council, U.K.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

---

aml *Remission Times for Acute Myelogenous Leukaemia*

Description

The `aml` data frame has 23 rows and 3 columns.

A clinical trial to evaluate the efficacy of maintenance chemotherapy for acute myelogenous leukaemia was conducted by Embury et al. (1977) at Stanford University. After reaching a stage of remission through treatment by chemotherapy, patients were randomized into two groups. The first group received maintenance chemotherapy and the second group did not. The aim of the study was to see if maintenance chemotherapy increased the length of the remission. The data here formed a preliminary analysis which was conducted in October 1974.

Usage

```
aml
```

Format

This data frame contains the following columns:

`time` The length of the complete remission (in weeks).

`cens` An indicator of right censoring. 1 indicates that the patient had a relapse and so `time` is the length of the remission. 0 indicates that the patient had left the study or was still in remission in October 1974, that is the length of remission is right-censored.

`group` The group into which the patient was randomized. Group 1 received maintenance chemotherapy, group 2 did not.

Note

Package **survival** also has a dataset `aml`. It is the same data with different names and with `group` replaced by a factor `x`.

Source

The data were obtained from

Miller, R.G. (1981) *Survival Analysis*. John Wiley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Embury, S.H, Elias, L., Heller, P.H., Hood, C.E., Greenberg, P.L. and Schrier, S.L. (1977) Remission maintenance therapy in acute myelogenous leukaemia. *Western Journal of Medicine*, **126**, 267-272.

`beaver` *Beaver Body Temperature Data*

Description

The `beaver` data frame has 100 rows and 4 columns. It is a multivariate time series of class `"ts"` and also inherits from class `"data.frame"`.

This data set is part of a long study into body temperature regulation in beavers. Four adult female beavers were live-trapped and had a temperature-sensitive radio transmitter surgically implanted. Readings were taken every 10 minutes. The location of the beaver was also recorded and her activity level was dichotomized by whether she was in the retreat or outside of it since high-intensity activities only occur outside of the retreat.

The data in this data frame are those readings for one of the beavers on a day in autumn.

Usage

```
beaver
```

Format

This data frame contains the following columns:

`day` The day number. The data includes only data from day 307 and early 308.

`time` The time of day formatted as hour-minute.

`temp` The body temperature in degrees Celsius.

`activ` The dichotomized activity indicator. `1` indicates that the beaver is outside of the retreat and therefore engaged in high-intensity activity.

Source

The data were obtained from

Reynolds, P.S. (1994) Time-series analyses of beaver body temperatures. In *Case Studies in Biometry*. N. Lange, L. Ryan, L. Billard, D. Brillinger, L. Conquest and J. Greenhouse (editors), 211–228. John Wiley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`bigcity` *Population of U.S. Cities*

Description

The `bigcity` data frame has 49 rows and 2 columns.

The `city` data frame has 10 rows and 2 columns.

The measurements are the population (in 1000's) of 49 U.S. cities in 1920 and 1930. The 49 cities are a random sample taken from the 196 largest cities in 1920. The `city` data frame consists of the first 10 observations in `bigcity`.

Usage

```
bigcity
```

Format

This data frame contains the following columns:

`u` The 1920 population.

`x` The 1930 population.

Source

The data were obtained from

Cochran, W.G. (1977) *Sampling Techniques*. Third edition. John Wiley

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`boot` *Bootstrap Resampling*

Description

Generate `R` bootstrap replicates of a statistic applied to data. Both parametric and nonparametric resampling are possible. For the nonparametric bootstrap, possible resampling methods are the ordinary bootstrap, the balanced bootstrap, antithetic resampling, and permutation. For nonparametric multi-sample problems stratified resampling is used: this is specified by including a vector of strata in the call to boot. Importance resampling weights may be specified.

Usage

```
boot(data, statistic, R, sim = "ordinary", stype = c("i", "f", "w"),
     strata = rep(1,n), L = NULL, m = 0, weights = NULL,
     ran.gen = function(d, p) d, mle = NULL, simple = FALSE, ...,
     parallel = c("no", "multicore", "snow"),
     ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

Arguments

`data`	The data as a vector, matrix or data frame. If it is a matrix or data frame then each row is considered as one multivariate observation.
`statistic`	A function which when applied to data returns a vector containing the statistic(s) of interest. When `sim = "parametric"`, the first argument to `statistic` must be the data. For each replicate a simulated dataset returned by `ran.gen` will be passed. In all other cases `statistic` must take at least two arguments. The first argument passed will always be the original data. The second will be a vector of indices, frequencies or weights which define the bootstrap sample. Further, if predictions are required, then a third argument is required which would be a vector of the random indices used to generate the bootstrap predictions. Any further arguments can be passed to `statistic` through the `...` argument.
`R`	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case `R` would be a vector of integers where each component gives the number of resamples from each of the rows of weights.

`sim` A character string indicating the type of simulation required. Possible values are `"ordinary"` (the default), `"parametric"`, `"balanced"`, `"permutation"`, or `"antithetic"`. Importance resampling is specified by including importance weights; the type of importance resampling must still be specified but may only be `"ordinary"` or `"balanced"` in this case.

`stype` A character string indicating what the second argument of `statistic` represents. Possible values of stype are `"i"` (indices - the default), `"f"` (frequencies), or `"w"` (weights). Not used for `sim = "parametric"`.

`strata` An integer vector or factor specifying the strata for multi-sample problems. This may be specified for any simulation, but is ignored when `sim = "parametric"`. When `strata` is supplied for a nonparametric bootstrap, the simulations are done within the specified strata.

`L` Vector of influence values evaluated at the observations. This is used only when `sim` is `"antithetic"`. If not supplied, they are calculated through a call to `empinf`. This will use the infinitesimal jackknife provided that `stype` is `"w"`, otherwise the usual jackknife is used.

`m` The number of predictions which are to be made at each bootstrap replicate. This is most useful for (generalized) linear models. This can only be used when `sim` is `"ordinary"`. `m` will usually be a single integer but, if there are strata, it may be a vector with length equal to the number of strata, specifying how many of the errors for prediction should come from each strata. The actual predictions should be returned as the final part of the output of `statistic`, which should also take an argument giving the vector of indices of the errors to be used for the predictions.

`weights` Vector or matrix of importance weights. If a vector then it should have as many elements as there are observations in `data`. When simulation from more than one set of weights is required, `weights` should be a matrix where each row of the matrix is one set of importance weights. If `weights` is a matrix then `R` must be a vector of length `nrow(weights)`. This parameter is ignored if `sim` is not `"ordinary"` or `"balanced"`.

`ran.gen` This function is used only when `sim = "parametric"` when it describes how random values are to be generated. It should be a function of two arguments. The first argument should be the observed data and the second argument consists of any other information needed (e.g. parameter estimates). The second argument may be a list, allowing any number of items to be passed to `ran.gen`. The returned value should be a simulated data set of the same form as the observed data which will be passed to `statistic` to get a bootstrap replicate. It is important that the returned value be of the same shape and type as the original dataset. If `ran.gen` is not specified, the default is a function which returns the original `data` in which case all simulation should be included as part of `statistic`. Use of `sim = "parametric"` with a suitable `ran.gen` allows the user to implement any types of nonparametric resampling which are not supported directly.

`mle` The second argument to be passed to `ran.gen`. Typically these will be maximum likelihood estimates of the parameters. For efficiency `mle` is often a list containing all of the objects needed by `ran.gen` which can be calculated using the original data set only.

simple	logical, only allowed to be TRUE for sim = "ordinary", stype = "i", n = 0 (otherwise ignored with a warning). By default a n by R index array is created: this can be large and if simple = TRUE this is avoided by sampling separately for each replication, which is slower but uses less memory.
...	Other named arguments for statistic which are passed unchanged each time it is called. Any such arguments to statistic should follow the arguments which statistic is required to have for the simulation. Beware of partial matching to arguments of boot listed above, and that arguments named X and FUN cause conflicts in some versions of **boot** (but not this one).
parallel	The type of parallel operation to be used (if any). If missing, the default is taken from the option "boot.parallel" (and if that is not set, "no").
ncpus	integer: number of processes to be used in parallel operation: typically one would chose this to the number of available CPUs.
cl	An optional **parallel** or **snow** cluster for use if parallel = "snow". If not supplied, a cluster on the local machine is created for the duration of the boot call.

Details

The statistic to be bootstrapped can be as simple or complicated as desired as long as its arguments correspond to the dataset and (for a nonparametric bootstrap) a vector of indices, frequencies or weights. statistic is treated as a black box by the boot function and is not checked to ensure that these conditions are met.

The first order balanced bootstrap is described in Davison, Hinkley and Schechtman (1986). The antithetic bootstrap is described by Hall (1989) and is experimental, particularly when used with strata. The other non-parametric simulation types are the ordinary bootstrap (possibly with unequal probabilities), and permutation which returns random permutations of cases. All of these methods work independently within strata if that argument is supplied.

For the parametric bootstrap it is necessary for the user to specify how the resampling is to be conducted. The best way of accomplishing this is to specify the function ran.gen which will return a simulated data set from the observed data set and a set of parameter estimates specified in mle.

Value

The returned value is an object of class "boot", containing the following components:

t0	The observed value of statistic applied to data.
t	A matrix with sum(R) rows each of which is a bootstrap replicate of the result of calling statistic.
R	The value of R as passed to boot.
data	The data as passed to boot.
seed	The value of .Random.seed when boot started work.
statistic	The function statistic as passed to boot.
sim	Simulation type used.
stype	Statistic type as passed to boot.

call	The original call to boot.
strata	The strata used. This is the vector passed to boot, if it was supplied or a vector of ones if there were no strata. It is not returned if sim is "parametric".
weights	The importance sampling weights as passed to boot or the empirical distribution function weights if no importance sampling weights were specified. It is omitted if sim is not one of "ordinary" or "balanced".
pred.i	If predictions are required (m > 0) this is the matrix of indices at which predictions were calculated as they were passed to statistic. Omitted if m is 0 or sim is not "ordinary".
L	The influence values used when sim is "antithetic". If no such values were specified and stype is not "w" then L is returned as consecutive integers corresponding to the assumption that data is ordered by influence values. This component is omitted when sim is not "antithetic".
ran.gen	The random generator function used if sim is "parametric". This component is omitted for any other value of sim.
mle	The parameter estimates passed to boot when sim is "parametric". It is omitted for all other values of sim.

There are c, plot and print methods for this class.

Parallel operation

When parallel = "multicore" is used (not available on Windows), each worker process inherits the environment of the current session, including the workspace and the loaded namespaces and attached packages (but not the random number seed: see below).

More work is needed when parallel = "snow" is used: the worker processes are newly created R processes, and statistic needs to arrange to set up the environment it needs: often a good way to do that is to make use of lexical scoping since when statistic is sent to the worker processes its enclosing environment is also sent. (E.g. see the example for jack.after.boot where ancillary functions are nested inside the statistic function.) parallel = "snow" is primarily intended to be used on multi-core Windows machine where parallel = "multicore" is not available.

For most of the boot methods the resampling is done in the master process, but not if simple = TRUE nor sim = "parametric". In those cases (or where statistic itself uses random numbers), more care is needed if the results need to be reproducible. Resampling is done in the worker processes by censboot(sim = "wierd") and by most of the schemes in tsboot (the exceptions being sim == "fixed" and sim == "geom" with the default ran.gen).

Where random-number generation is done in the worker processes, the default behaviour is that each worker chooses a separate seed, non-reproducibly. However, with parallel = "multicore" or parallel = "snow" using the default cluster, a second approach is used if RNGkind("L'Ecuyer-CMRG") has been selected. In that approach each worker gets a different subsequence of the RNG stream based on the seed at the time the worker is spawned and so the results will be reproducible if ncpus is unchanged, and for parallel = "multicore" if parallel::mc.reset.stream() is called: see the examples for mclapply.

Note that loading the **parallel** namespace may change the random seed, so for maximum reproducibility this should be done before calling this function.

References

There are many references explaining the bootstrap and its variations. Among them are :

Booth, J.G., Hall, P. and Wood, A.T.A. (1993) Balanced importance resampling for the bootstrap. *Annals of Statistics*, **21**, 286–298.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Davison, A.C., Hinkley, D.V. and Schechtman, E. (1986) Efficient bootstrap simulation. *Biometrika*, **73**, 555–566.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman & Hall.

Gleason, J.R. (1988) Algorithms for balanced bootstrap simulations. *American Statistician*, **42**, 263–266.

Hall, P. (1989) Antithetic resampling for the bootstrap. *Biometrika*, **73**, 713–724.

Hinkley, D.V. (1988) Bootstrap methods (with Discussion). *Journal of the Royal Statistical Society, B*, **50**, 312–337, 355–370.

Hinkley, D.V. and Shi, S. (1989) Importance sampling and the nested bootstrap. *Biometrika*, **76**, 435–446.

Johns M.V. (1988) Importance sampling for bootstrap confidence intervals. *Journal of the American Statistical Association*, **83**, 709–714.

Noreen, E.W. (1989) *Computer Intensive Methods for Testing Hypotheses*. John Wiley & Sons.

See Also

`boot.array`, `boot.ci`, `censboot`, `empinf`, `jack.after.boot`, `tilt.boot`, `tsboot`.

Examples

```
# Usual bootstrap of the ratio of means using the city data
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
boot(city, ratio, R = 999, stype = "w")


# Stratified resampling for the difference of means.  In this
# example we will look at the difference of means between the final
# two series in the gravity data.
diff.means <- function(d, f)
{    n <- nrow(d)
     gp1 <- 1:table(as.numeric(d$series))[1]
     m1 <- sum(d[gp1,1] * f[gp1])/sum(f[gp1])
     m2 <- sum(d[-gp1,1] * f[-gp1])/sum(f[-gp1])
     ss1 <- sum(d[gp1,1]^2 * f[gp1]) - (m1 *  m1 * sum(f[gp1]))
     ss2 <- sum(d[-gp1,1]^2 * f[-gp1]) - (m2 *  m2 * sum(f[-gp1]))
     c(m1 - m2, (ss1 + ss2)/(sum(f) - 2))
}
grav1 <- gravity[as.numeric(gravity[,2]) >= 7,]
boot(grav1, diff.means, R = 999, stype = "f", strata = grav1[,2])
```

```
# In this example we show the use of boot in a prediction from
# regression based on the nuclear data.  This example is taken
# from Example 6.8 of Davison and Hinkley (1997).  Notice also
# that two extra arguments to 'statistic' are passed through boot.
nuke <- nuclear[, c(1, 2, 5, 7, 8, 10, 11)]
nuke.lm <- glm(log(cost) ~ date+log(cap)+ne+ct+log(cum.n)+pt, data = nuke)
nuke.diag <- glm.diag(nuke.lm)
nuke.res <- nuke.diag$res * nuke.diag$sd
nuke.res <- nuke.res - mean(nuke.res)

# We set up a new data frame with the data, the standardized
# residuals and the fitted values for use in the bootstrap.
nuke.data <- data.frame(nuke, resid = nuke.res, fit = fitted(nuke.lm))

# Now we want a prediction of plant number 32 but at date 73.00
new.data <- data.frame(cost = 1, date = 73.00, cap = 886, ne = 0,
                       ct = 0, cum.n = 11, pt = 1)
new.fit <- predict(nuke.lm, new.data)

nuke.fun <- function(dat, inds, i.pred, fit.pred, x.pred)
{
     lm.b <- glm(fit+resid[inds] ~ date+log(cap)+ne+ct+log(cum.n)+pt,
                 data = dat)
     pred.b <- predict(lm.b, x.pred)
     c(coef(lm.b), pred.b - (fit.pred + dat$resid[i.pred]))
}

nuke.boot <- boot(nuke.data, nuke.fun, R = 999, m = 1,
                  fit.pred = new.fit, x.pred = new.data)
# The bootstrap prediction squared error would then be found by
mean(nuke.boot$t[, 8]^2)
# Basic bootstrap prediction limits would be
new.fit - sort(nuke.boot$t[, 8])[c(975, 25)]


# Finally a parametric bootstrap.  For this example we shall look
# at the air-conditioning data.  In this example our aim is to test
# the hypothesis that the true value of the index is 1 (i.e. that
# the data come from an exponential distribution) against the
# alternative that the data come from a gamma distribution with
# index not equal to 1.
air.fun <- function(data) {
     ybar <- mean(data$hours)
     para <- c(log(ybar), mean(log(data$hours)))
     ll <- function(k) {
          if (k <= 0) 1e200 else lgamma(k)-k*(log(k)-1-para[1]+para[2])
     }
     khat <- nlm(ll, ybar^2/var(data$hours))$estimate
     c(ybar, khat)
}

air.rg <- function(data, mle) {
    # Function to generate random exponential variates.
```

```
    # mle will contain the mean of the original data
    out <- data
    out$hours <- rexp(nrow(out), 1/mle)
    out
}

air.boot <- boot(aircondit, air.fun, R = 999, sim = "parametric",
                 ran.gen = air.rg, mle = mean(aircondit$hours))

# The bootstrap p-value can then be approximated by
sum(abs(air.boot$t[,2]-1) > abs(air.boot$t0[2]-1))/(1+air.boot$R)
```

`boot.array` *Bootstrap Resampling Arrays*

Description

This function takes a bootstrap object calculated by one of the functions `boot`, `censboot`, or `tilt.boot` and returns the frequency (or index) array for the bootstrap resamples.

Usage

```
boot.array(boot.out, indices)
```

Arguments

`boot.out`	An object of class `"boot"` returned by one of the generation functions for such an object.
`indices`	A logical argument which specifies whether to return the frequency array or the raw index array. The default is `indices=FALSE` unless `boot.out` was created by `tsboot` in which case the default is `indices=TRUE`.

Details

The process by which the original index array was generated is repeated with the same value of `.Random.seed`. If the frequency array is required then `freq.array` is called to convert the index array to a frequency array.

A resampling array can only be returned when such a concept makes sense. In particular it cannot be found for any parametric or model-based resampling schemes. Hence for objects generated by `censboot` the only resampling scheme for which such an array can be found is ordinary case resampling. Similarly if `boot.out$sim` is `"parametric"` in the case of `boot` or `"model"` in the case of `tsboot` the array cannot be found. Note also that for post-blackened bootstraps from `tsboot` the indices found will relate to those prior to any post-blackening and so will not be useful.

Frequency arrays are used in many post-bootstrap calculations such as the jackknife-after-bootstrap and finding importance sampling weights. They are also used to find empirical influence values through the regression method.

Value

A matrix with `boot.out$R` rows and `n` columns where `n` is the number of observations in `boot.out$data`. If `indices` is `FALSE` then this will give the frequency of each of the original observations in each bootstrap resample. If `indices` is `TRUE` it will give the indices of the bootstrap resamples in the order in which they would have been passed to the statistic.

Side Effects

This function temporarily resets `.Random.seed` to the value in `boot.out$seed` and then returns it to its original value at the end of the function.

See Also

`boot`, `censboot`, `freq.array`, `tilt.boot`, `tsboot`

Examples

```
#  A frequency array for a nonparametric bootstrap
city.boot <- boot(city, corr, R = 40, stype = "w")
boot.array(city.boot)

perm.cor <- function(d,i) cor(d$x,d$u[i])
city.perm <- boot(city, perm.cor, R = 40, sim = "permutation")
boot.array(city.perm, indices = TRUE)
```

---

boot.ci *Nonparametric Bootstrap Confidence Intervals*

Description

This function generates 5 different types of equi-tailed two-sided nonparametric confidence intervals. These are the first order normal approximation, the basic bootstrap interval, the studentized bootstrap interval, the bootstrap percentile interval, and the adjusted bootstrap percentile (BCa) interval. All or a subset of these intervals can be generated.

Usage

```
boot.ci(boot.out, conf = 0.95, type = "all",
        index = 1:min(2,length(boot.out$t0)), var.t0 = NULL,
        var.t = NULL, t0 = NULL, t = NULL, L = NULL,
        h = function(t) t, hdot = function(t) rep(1,length(t)),
        hinv = function(t) t, ...)
```

Arguments

boot.out	An object of class "boot" containing the output of a bootstrap calculation.
conf	A scalar or vector containing the confidence level(s) of the required interval(s).
type	A vector of character strings representing the type of intervals required. The value should be any subset of the values c("norm","basic", "stud", "perc", "bca") or simply "all" which will compute all five types of intervals.
index	This should be a vector of length 1 or 2. The first element of index indicates the position of the variable of interest in boot.out$t0 and the relevant column in boot.out$t. The second element indicates the position of the variance of the variable of interest. If both var.t0 and var.t are supplied then the second element of index (if present) is ignored. The default is that the variable of interest is in position 1 and its variance is in position 2 (as long as there are 2 positions in boot.out$t0).
var.t0	If supplied, a value to be used as an estimate of the variance of the statistic for the normal approximation and studentized intervals. If it is not supplied and length(index) is 2 then var.t0 defaults to boot.out$t0[index[2]] otherwise var.t0 is undefined. For studentized intervals var.t0 must be defined. For the normal approximation, if var.t0 is undefined it defaults to var(t). If a transformation is supplied through the argument h then var.t0 should be the variance of the untransformed statistic.
var.t	This is a vector (of length boot.out$R) of variances of the bootstrap replicates of the variable of interest. It is used only for studentized intervals. If it is not supplied and length(index) is 2 then var.t defaults to boot.out$t[,index[2]], otherwise its value is undefined which will cause an error for studentized intervals. If a transformation is supplied through the argument h then var.t should be the variance of the untransformed bootstrap statistics.
t0	The observed value of the statistic of interest. The default value is boot.out$t0[index[1]]. Specification of t0 and t allows the user to get intervals for a transformed statistic which may not be in the bootstrap output object. See the second example below. An alternative way of achieving this would be to supply the functions h, hdot, and hinv below.
t	The bootstrap replicates of the statistic of interest. It must be a vector of length boot.out$R. It is an error to supply one of t0 or t but not the other. Also if studentized intervals are required and t0 and t are supplied then so should be var.t0 and var.t. The default value is boot.out$t[,index].
L	The empirical influence values of the statistic of interest for the observed data. These are used only for BCa intervals. If a transformation is supplied through the parameter h then L should be the influence values for t; the values for h(t) are derived from these and hdot within the function. If L is not supplied then the values are calculated using empinf if they are needed.
h	A function defining a transformation. The intervals are calculated on the scale of h(t) and the inverse function hinv applied to the resulting intervals. It must be a function of one variable only and for a vector argument, it must return a vector of the same length, i.e. h(c(t1,t2,t3)) should return c(h(t1),h(t2),h(t3)). The default is the identity function.

hdot	A function of one argument returning the derivative of h. It is a required argument if h is supplied and normal, studentized or BCa intervals are required. The function is used for approximating the variances of h(t0) and h(t) using the delta method, and also for finding the empirical influence values for BCa intervals. Like h it should be able to take a vector argument and return a vector of the same length. The default is the constant function 1.
hinv	A function, like h, which returns the inverse of h. It is used to transform the intervals calculated on the scale of h(t) back to the original scale. The default is the identity function. If h is supplied but hinv is not, then the intervals returned will be on the transformed scale.
...	Any extra arguments that boot.out$statistic is expecting. These arguments are needed only if BCa intervals are required and L is not supplied since in that case L is calculated through a call to empinf which calls boot.out$statistic.

Details

The formulae on which the calculations are based can be found in Chapter 5 of Davison and Hinkley (1997). Function boot must be run prior to running this function to create the object to be passed as boot.out.

Variance estimates are required for studentized intervals. The variance of the observed statistic is optional for normal theory intervals. If it is not supplied then the bootstrap estimate of variance is used. The normal intervals also use the bootstrap bias correction.

Interpolation on the normal quantile scale is used when a non-integer order statistic is required. If the order statistic used is the smallest or largest of the R values in boot.out a warning is generated and such intervals should not be considered reliable.

Value

An object of type "bootci" which contains the intervals. It has components

R	The number of bootstrap replicates on which the intervals were based.
t0	The observed value of the statistic on the same scale as the intervals.
call	The call to boot.ci which generated the object.
	It will also contain one or more of the following components depending on the value of type used in the call to bootci.
normal	A matrix of intervals calculated using the normal approximation. It will have 3 columns, the first being the level and the other two being the upper and lower endpoints of the intervals.
basic	The intervals calculated using the basic bootstrap method.
student	The intervals calculated using the studentized bootstrap method.
percent	The intervals calculated using the bootstrap percentile method.
bca	The intervals calculated using the adjusted bootstrap percentile (BCa) method.
	These latter four components will be matrices with 5 columns, the first column containing the level, the next two containing the indices of the order statistics used in the calculations and the final two the calculated endpoints themselves.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*, Chapter 5. Cambridge University Press.

DiCiccio, T.J. and Efron B. (1996) Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.

Efron, B. (1987) Better bootstrap confidence intervals (with Discussion). *Journal of the American Statistical Association*, **82**, 171–200.

See Also

abc.ci, boot, empinf, norm.ci

Examples

```
# confidence intervals for the city data
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
city.boot <- boot(city, ratio, R = 999, stype = "w", sim = "ordinary")
boot.ci(city.boot, conf = c(0.90, 0.95),
        type = c("norm", "basic", "perc", "bca"))

# studentized confidence interval for the two sample
# difference of means problem using the final two series
# of the gravity data.
diff.means <- function(d, f)
{    n <- nrow(d)
     gp1 <- 1:table(as.numeric(d$series))[1]
     m1 <- sum(d[gp1,1] * f[gp1])/sum(f[gp1])
     m2 <- sum(d[-gp1,1] * f[-gp1])/sum(f[-gp1])
     ss1 <- sum(d[gp1,1]^2 * f[gp1]) - (m1 *  m1 * sum(f[gp1]))
     ss2 <- sum(d[-gp1,1]^2 * f[-gp1]) - (m2 *  m2 * sum(f[-gp1]))
     c(m1 - m2, (ss1 + ss2)/(sum(f) - 2))
}
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav1.boot <- boot(grav1, diff.means, R = 999, stype = "f",
                   strata = grav1[ ,2])
boot.ci(grav1.boot, type = c("stud", "norm"))

# Nonparametric confidence intervals for mean failure time
# of the air-conditioning data as in Example 5.4 of Davison
# and Hinkley (1997)
mean.fun <- function(d, i)
{    m <- mean(d$hours[i])
     n <- length(i)
     v <- (n-1)*var(d$hours[i])/n^2
     c(m, v)
}
air.boot <- boot(aircondit, mean.fun, R = 999)
boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"))

# Now using the log transformation
# There are two ways of doing this and they both give the
```

```
# same intervals.

# Method 1
boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"),
        h = log, hdot = function(x) 1/x)

# Method 2
vt0 <- air.boot$t0[2]/air.boot$t0[1]^2
vt <- air.boot$t[, 2]/air.boot$t[ ,1]^2
boot.ci(air.boot, type = c("norm", "basic", "perc", "stud"),
        t0 = log(air.boot$t0[1]), t = log(air.boot$t[,1]),
        var.t0 = vt0, var.t = vt)
```

`brambles` *Spatial Location of Bramble Canes*

Description

The `brambles` data frame has 823 rows and 3 columns.

The location of living bramble canes in a 9m square plot was recorded. We take 9m to be the unit of distance so that the plot can be thought of as a unit square. The bramble canes were also classified by their age.

Usage

```
brambles
```

Format

This data frame contains the following columns:

`x` The x coordinate of the position of the cane in the plot.

`y` The y coordinate of the position of the cane in the plot.

`age` The age classification of the canes; `0` indicates a newly emerged cane, `1` indicates a one year old cane and `2` indicates a two year old cane.

Source

The data were obtained from

Diggle, P.J. (1983) *Statistical Analysis of Spatial Point Patterns*. Academic Press.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`breslow` *Smoking Deaths Among Doctors*

Description

The `breslow` data frame has 10 rows and 5 columns.

In 1961 Doll and Hill sent out a questionnaire to all men on the British Medical Register enquiring about their smoking habits. Almost 70% of such men replied. Death certificates were obtained for medical practitioners and causes of death were assigned on the basis of these certificates. The `breslow` data set contains the person-years of observations and deaths from coronary artery disease accumulated during the first ten years of the study.

Usage

```
breslow
```

Format

This data frame contains the following columns:

`age` The mid-point of the 10 year age-group for the doctors.

`smoke` An indicator of whether the doctors smoked (1) or not (0).

`n` The number of person-years in the category.

`y` The number of deaths attributed to coronary artery disease.

`ns` The number of smoker years in the category (`smoke*n`).

Source

The data were obtained from

Breslow, N.E. (1985) Cohort Analysis in Epidemiology. In *A Celebration of Statistics* A.C. Atkinson and S.E. Fienberg (editors), 109–143. Springer-Verlag.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Doll, R. and Hill, A.B. (1966) Mortality of British doctors in relation to smoking: Observations on coronary thrombosis. *National Cancer Institute Monograph*, **19**, 205-268.

calcium *Calcium Uptake Data*

Description

The calcium data frame has 27 rows and 2 columns.

Howard Grimes from the Botany Department, North Carolina State University, conducted an experiment for biochemical analysis of intracellular storage and transport of calcium across plasma membrane. Cells were suspended in a solution of radioactive calcium for a certain length of time and then the amount of radioactive calcium that was absorbed by the cells was measured. The experiment was repeated independently with 9 different times of suspension each replicated 3 times.

Usage

```
calcium
```

Format

This data frame contains the following columns:

time The time (in minutes) that the cells were suspended in the solution.

cal The amount of calcium uptake (nmoles/mg).

Source

The data were obtained from

Rawlings, J.O. (1988) *Applied Regression Analysis*. Wadsworth and Brooks/Cole Statistics/Probability Series.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

cane *Sugar-cane Disease Data*

Description

The cane data frame has 180 rows and 5 columns. The data frame represents a randomized block design with 45 varieties of sugar-cane and 4 blocks.

Usage

```
cane
```

Format

This data frame contains the following columns:

`n` The total number of shoots in each plot.

`r` The number of diseased shoots.

`x` The number of pieces of the stems, out of 50, planted in each plot.

`var` A factor indicating the variety of sugar-cane in each plot.

`block` A factor for the blocks.

Details

The aim of the experiment was to classify the varieties into resistant, intermediate and susceptible to a disease called "coal of sugar-cane" (carvao da cana-de-acucar). This is a disease that is common in sugar-cane plantations in certain areas of Brazil.

For each plot, fifty pieces of sugar-cane stem were put in a solution containing the disease agent and then some were planted in the plot. After a fixed period of time, the total number of shoots and the number of diseased shoots were recorded.

Source

The data were kindly supplied by Dr. C.G.B. Demetrio of Escola Superior de Agricultura, Universidade de Sao Paolo, Brazil.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`capability` *Simulated Manufacturing Process Data*

Description

The `capability` data frame has 75 rows and 1 columns.

The data are simulated successive observations from a process in equilibrium. The process is assumed to have specification limits (5.49, 5.79).

Usage

```
capability
```

Format

This data frame contains the following column:

`y` The simulated measurements.

Source

The data were obtained from

Bissell, A.F. (1990) How reliable is your capability index? *Applied Statistics*, **39**, 331–340.

References

Canty, A.J. and Davison, A.C. (1996) Implementation of saddlepoint approximations to resampling distributions. To appear in *Computing Science and Statistics; Proceedings of the 28th Symposium on the Interface*.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`catsM` *Weight Data for Domestic Cats*

Description

The `catsM` data frame has 97 rows and 3 columns.

144 adult (over 2kg in weight) cats used for experiments with the drug digitalis had their heart and body weight recorded. 47 of the cats were female and 97 were male. The `catsM` data frame consists of the data for the male cats. The full data are in dataset `cats` in package `MASS`.

Usage

```
cats
```

Format

This data frames contain the following columns:

`Sex` A factor for the sex of the cat (levels are `F` and `M`).

`Bwt` Body weight in kg.

`Hwt` Heart weight in g.

Source

The data were obtained from

Fisher, R.A. (1947) The analysis of covariance method for the relation between a part and the whole. *Biometrics*, **3**, 65–68.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Venables, W.N. and Ripley, B.D. (1994) *Modern Applied Statistics with S-Plus*. Springer-Verlag.

See Also

cats

---

cav *Position of Muscle Caveolae*

Description

The cav data frame has 138 rows and 2 columns.

The data gives the positions of the individual caveolae in a square region with sides of length 500 units. This grid was originally on a 2.65mum square of muscle fibre. The data are those points falling in the lower left hand quarter of the region used for the dataset caveolae.dat in the **spatial** package by B.D. Ripley (1994).

Usage

```
cav
```

Format

This data frame contains the following columns:

x The x coordinate of the caveola's position in the region.

y The y coordinate of the caveola's position in the region.

References

Appleyard, S.T., Witkowski, J.A., Ripley, B.D., Shotton, D.M. and Dubowicz, V. (1985) A novel procedure for pattern analysis of features present on freeze fractured plasma membranes. *Journal of Cell Science*, **74**, 105–117.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

---

cd4 *CD4 Counts for HIV-Positive Patients*

Description

The cd4 data frame has 20 rows and 2 columns.

CD4 cells are carried in the blood as part of the human immune system. One of the effects of the HIV virus is that these cells die. The count of CD4 cells is used in determining the onset of full-blown AIDS in a patient. In this study of the effectiveness of a new anti-viral drug on HIV, 20 HIV-positive patients had their CD4 counts recorded and then were put on a course of treatment with this drug. After using the drug for one year, their CD4 counts were again recorded. The aim of the experiment was to show that patients taking the drug had increased CD4 counts which is not generally seen in HIV-positive patients.

Usage

```
cd4
```

Format

This data frame contains the following columns:

`baseline` The CD4 counts (in 100's) on admission to the trial.

`oneyear` The CD4 counts (in 100's) after one year of treatment with the new drug.

Source

The data were obtained from

DiCiccio, T.J. and Efron B. (1996) Bootstrap confidence intervals (with Discussion). *Statistical Science*, **11**, 189–228.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

---

cd4.nested *Nested Bootstrap of cd4 data*

Description

This is an example of a nested bootstrap for the correlation coefficient of the `cd4` data frame. It is used in a practical in Chapter 5 of Davison and Hinkley (1997).

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

`cd4`

censboot *Bootstrap for Censored Data*

Description

This function applies types of bootstrap resampling which have been suggested to deal with right-censored data. It can also do model-based resampling using a Cox regression model.

Usage

```
censboot(data, statistic, R, F.surv, G.surv, strata = matrix(1,n,2),
         sim = "ordinary", cox = NULL, index = c(1, 2), ...,
         parallel = c("no", "multicore", "snow"),
         ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

Arguments

data	The data frame or matrix containing the data. It must have at least two columns, one of which contains the times and the other the censoring indicators. It is allowed to have as many other columns as desired (although efficiency is reduced for large numbers of columns) except for `sim = "weird"` when it should only have two columns - the times and censoring indicators. The columns of `data` referenced by the components of `index` are taken to be the times and censoring indicators.
statistic	A function which operates on the data frame and returns the required statistic. Its first argument must be the data. Any other arguments that it requires can be passed using the `...` argument. In the case of `sim = "weird"`, the data passed to `statistic` only contains the times and censoring indicator regardless of the actual number of columns in `data`. In all other cases the data passed to statistic will be of the same form as the original data. When `sim = "weird"`, the actual number of observations in the resampled data sets may not be the same as the number in `data`. For this reason, if `sim = "weird"` and `strata` is supplied, `statistic` should also take a numeric vector indicating the strata. This allows the statistic to depend on the strata if required.
R	The number of bootstrap replicates.
F.surv	An object returned from a call to `survfit` giving the survivor function for the data. This is a required argument unless `sim = "ordinary"` or `sim = "model"` and `cox` is missing.
G.surv	Another object returned from a call to `survfit` but with the censoring indicators reversed to give the product-limit estimate of the censoring distribution. Note that for consistency the uncensored times should be reduced by a small amount in the call to `survfit`. This is a required argument whenever `sim = "cond"` or when `sim = "model"` and `cox` is supplied.
strata	The strata used in the calls to `survfit`. It can be a vector or a matrix with 2 columns. If it is a vector then it is assumed to be the strata for the survival

distribution, and the censoring distribution is assumed to be the same for all observations. If it is a matrix then the first column is the strata for the survival distribution and the second is the strata for the censoring distribution. When `sim = "weird"` only the strata for the survival distribution are used since the censoring times are considered fixed. When `sim = "ordinary"`, only one set of strata is used to stratify the observations, this is taken to be the first column of `strata` when it is a matrix.

`sim` The simulation type. Possible types are `"ordinary"` (case resampling), `"model"` (equivalent to `"ordinary"` if `cox` is missing, otherwise it is model-based resampling), `"weird"` (the weird bootstrap - this cannot be used if `cox` is supplied), and `"cond"` (the conditional bootstrap, in which censoring times are resampled from the conditional censoring distribution).

`cox` An object returned from `coxph`. If it is supplied, then `F.surv` should have been generated by a call of the form `survfit(cox)`.

`index` A vector of length two giving the positions of the columns in `data` which correspond to the times and censoring indicators respectively.

`...` Other named arguments which are passed unchanged to `statistic` each time it is called. Any such arguments to `statistic` must follow the arguments which `statistic` is required to have for the simulation. Beware of partial matching to arguments of `censboot` listed above, and that arguments named `X` and `FUN` cause conflicts in some versions of **boot** (but not this one).

`parallel, ncpus, cl`
See the help for `boot`.

Details

The various types of resampling are described in Davison and Hinkley (1997) in sections 3.5 and 7.3. The simplest is case resampling which simply resamples with replacement from the observations.

The conditional bootstrap simulates failure times from the estimate of the survival distribution. Then, for each observation its simulated censoring time is equal to the observed censoring time if the observation was censored and generated from the estimated censoring distribution conditional on being greater than the observed failure time if the observation was uncensored. If the largest value is censored then it is given a nominal failure time of `Inf` and conversely if it is uncensored it is given a nominal censoring time of `Inf`. This is necessary to allow the largest observation to be in the resamples.

If a Cox regression model is fitted to the data and supplied, then the failure times are generated from the survival distribution using that model. In this case the censoring times can either be simulated from the estimated censoring distribution (`sim = "model"`) or from the conditional censoring distribution as in the previous paragraph (`sim = "cond"`).

The weird bootstrap holds the censored observations as fixed and also the observed failure times. It then generates the number of events at each failure time using a binomial distribution with mean 1 and denominator the number of failures that could have occurred at that time in the original data set. In our implementation we insist that there is a least one simulated event in each stratum for every bootstrap dataset.

When there are strata involved and `sim` is either `"model"` or `"cond"` the situation becomes more difficult. Since the strata for the survival and censoring distributions are not the same it is possible

that for some observations both the simulated failure time and the simulated censoring time are infinite. To see this consider an observation in stratum 1F for the survival distribution and stratum 1G for the censoring distribution. Now if the largest value in stratum 1F is censored it is given a nominal failure time of `Inf`, also if the largest value in stratum 1G is uncensored it is given a nominal censoring time of `Inf` and so both the simulated failure and censoring times could be infinite. When this happens the simulated value is considered to be a failure at the time of the largest observed failure time in the stratum for the survival distribution.

When `parallel = "snow"` and `cl` is not supplied, `library(survival)` is run in each of the worker processes.

Value

An object of class `"boot"` containing the following components:

`t0`	The value of `statistic` when applied to the original data.
`t`	A matrix of bootstrap replicates of the values of `statistic`.
`R`	The number of bootstrap replicates performed.
`sim`	The simulation type used. This will usually be the input value of `sim` unless that was `"model"` but `cox` was not supplied, in which case it will be `"ordinary"`.
`data`	The data used for the bootstrap. This will generally be the input value of `data` unless `sim = "weird"`, in which case it will just be the columns containing the times and the censoring indicators.
`seed`	The value of `.Random.seed` when `censboot` started work.
`statistic`	The input value of `statistic`.
`strata`	The strata used in the resampling. When `sim = "ordinary"` this will be a vector which stratifies the observations, when `sim = "weird"` it is the strata for the survival distribution and in all other cases it is a matrix containing the strata for the survival distribution and the censoring distribution.
`call`	The original call to `censboot`.

Author(s)

Angelo J. Canty. Parallel extensions by Brian Ripley

References

Andersen, P.K., Borgan, O., Gill, R.D. and Keiding, N. (1993) *Statistical Models Based on Counting Processes*. Springer-Verlag.

Burr, D. (1994) A comparison of certain bootstrap confidence intervals in the Cox model. *Journal of the American Statistical Association*, **89**, 1290–1302.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1981) Censored data and the bootstrap. *Journal of the American Statistical Association*, **76**, 312–319.

Hjort, N.L. (1985) Bootstrapping Cox's regression model. Technical report NSF-241, Dept. of Statistics, Stanford University.

See Also

boot, coxph, survfit

Examples

```
library(survival)
# Example 3.9 of Davison and Hinkley (1997) does a bootstrap on some
# remission times for patients with a type of leukaemia.  The patients
# were divided into those who received maintenance chemotherapy and
# those who did not.  Here we are interested in the median remission
# time for the two groups.
data(aml, package = "boot") # not the version in survival.
aml.fun <- function(data) {
     surv <- survfit(Surv(time, cens) ~ group, data = data)
     out <- NULL
     st <- 1
     for (s in 1:length(surv$strata)) {
          inds <- st:(st + surv$strata[s]-1)
          md <- min(surv$time[inds[1-surv$surv[inds] >= 0.5]])
          st <- st + surv$strata[s]
          out <- c(out, md)
     }
     out
}
aml.case <- censboot(aml, aml.fun, R = 499, strata = aml$group)

# Now we will look at the same statistic using the conditional
# bootstrap and the weird bootstrap.  For the conditional bootstrap
# the survival distribution is stratified but the censoring
# distribution is not.

aml.s1 <- survfit(Surv(time, cens) ~ group, data = aml)
aml.s2 <- survfit(Surv(time-0.001*cens, 1-cens) ~ 1, data = aml)
aml.cond <- censboot(aml, aml.fun, R = 499, strata = aml$group,
     F.surv = aml.s1, G.surv = aml.s2, sim = "cond")


# For the weird bootstrap we must redefine our function slightly since
# the data will not contain the group number.
aml.fun1 <- function(data, str) {
     surv <- survfit(Surv(data[, 1], data[, 2]) ~ str)
     out <- NULL
     st <- 1
     for (s in 1:length(surv$strata)) {
          inds <- st:(st + surv$strata[s] - 1)
          md <- min(surv$time[inds[1-surv$surv[inds] >= 0.5]])
          st <- st + surv$strata[s]
          out <- c(out, md)
     }
     out
}
aml.wei <- censboot(cbind(aml$time, aml$cens), aml.fun1, R = 499,
```

```
     strata = aml$group,  F.surv = aml.s1, sim = "weird")

# Now for an example where a cox regression model has been fitted
# the data we will look at the melanoma data of Example 7.6 from
# Davison and Hinkley (1997).  The fitted model assumes that there
# is a different survival distribution for the ulcerated and
# non-ulcerated groups but that the thickness of the tumour has a
# common effect.  We will also assume that the censoring distribution
# is different in different age groups.  The statistic of interest
# is the linear predictor.  This is returned as the values at a
# number of equally spaced points in the range of interest.
data(melanoma, package = "boot")
library(splines)# for ns
mel.cox <- coxph(Surv(time, status == 1) ~ ns(thickness, df=4) + strata(ulcer),
                 data = melanoma)
mel.surv <- survfit(mel.cox)
agec <- cut(melanoma$age, c(0, 39, 49, 59, 69, 100))
mel.cens <- survfit(Surv(time - 0.001*(status == 1), status != 1) ~
                    strata(agec), data = melanoma)
mel.fun <- function(d) {
     t1 <- ns(d$thickness, df=4)
     cox <- coxph(Surv(d$time, d$status == 1) ~ t1+strata(d$ulcer))
     ind <- !duplicated(d$thickness)
     u <- d$thickness[!ind]
     eta <- cox$linear.predictors[!ind]
     sp <- smooth.spline(u, eta, df=20)
     th <- seq(from = 0.25, to = 10, by = 0.25)
     predict(sp, th)$y
}
mel.str <- cbind(melanoma$ulcer, agec)

# this is slow!
mel.mod <- censboot(melanoma, mel.fun, R = 499, F.surv = mel.surv,
     G.surv = mel.cens, cox = mel.cox, strata = mel.str, sim = "model")
# To plot the original predictor and a 95% pointwise envelope for it
mel.env <- envelope(mel.mod)$point
th <- seq(0.25, 10, by = 0.25)
plot(th, mel.env[1, ],  ylim = c(-2, 2),
     xlab = "thickness (mm)", ylab = "linear predictor", type = "n")
lines(th, mel.mod$t0, lty = 1)
matlines(th, t(mel.env), lty = 2)
```

---

channing *Channing House Data*

Description

The channing data frame has 462 rows and 5 columns.

Channing House is a retirement centre in Palo Alto, California. These data were collected between the opening of the house in 1964 until July 1, 1975. In that time 97 men and 365 women passed

through the centre. For each of these, their age on entry and also on leaving or death was recorded. A large number of the observations were censored mainly due to the resident being alive on July 1, 1975 when the data was collected. Over the time of the study 130 women and 46 men died at Channing House. Differences between the survival of the sexes, taking age into account, was one of the primary concerns of this study.

Usage

```
channing
```

Format

This data frame contains the following columns:

`sex` A factor for the sex of each resident (`"Male"` or `"Female"`).

`entry` The residents age (in months) on entry to the centre

`exit` The age (in months) of the resident on death, leaving the centre or July 1, 1975 whichever event occurred first.

`time` The length of time (in months) that the resident spent at Channing House. (`time=exit-entry`)

`cens` The indicator of right censoring. 1 indicates that the resident died at Channing House, 0 indicates that they left the house prior to July 1, 1975 or that they were still alive and living in the centre at that date.

Source

The data were obtained from

Hyde, J. (1980) Testing survival with incomplete observations. *Biostatistics Casebook*. R.G. Miller, B. Efron, B.W. Brown and L.E. Moses (editors), 31–46. John Wiley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`claridge` *Genetic Links to Left-handedness*

Description

The `claridge` data frame has 37 rows and 2 columns.

The data are from an experiment which was designed to look for a relationship between a certain genetic characteristic and handedness. The 37 subjects were women who had a son with mental retardation due to inheriting a defective X-chromosome. For each such mother a genetic measurement of their DNA was made. Larger values of this measurement are known to be linked to the defective gene and it was hypothesized that larger values might also be linked to a progressive shift away from right-handednesss. Each woman also filled in a questionnaire regarding which hand they

used for various tasks. From these questionnaires a measure of hand preference was found for each mother. The scale of this measure goes from 1, indicating someone who always favours their right hand, to 8, indicating someone who always favours their left hand. Between these two extremes are people who favour one hand for some tasks and the other for other tasks.

Usage

```
claridge
```

Format

This data frame contains the following columns:

`dnan` The genetic measurement on each woman's DNA.

`hand` The measure of left-handedness on an integer scale from 1 to 8.

Source

The data were kindly made available by Dr. Gordon S. Claridge from the Department of Experimental Psychology, University of Oxford.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

---

cloth *Number of Flaws in Cloth*

Description

The `cloth` data frame has 32 rows and 2 columns.

Usage

```
cloth
```

Format

This data frame contains the following columns:

`x` The length of the roll of cloth.

`y` The number of flaws found in the roll.

Source

The data were obtained from

Bissell, A.F. (1972) A negative binomial model with varying element size. *Biometrika*, **59**, 435–441.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`co.transfer` *Carbon Monoxide Transfer*

Description

The `co.transfer` data frame has 7 rows and 2 columns. Seven smokers with chickenpox had their levels of carbon monoxide transfer measured on entry to hospital and then again after 1 week. The main question being whether one week of hospitalization has changed the carbon monoxide transfer factor.

Usage

```
co.transfer
```

Format

This data frame contains the following columns:

`entry` Carbon monoxide transfer factor on entry to hospital.

`week` Carbon monoxide transfer one week after admittance to hospital.

Source

The data were obtained from

Hand, D.J., Daly, F., Lunn, A.D., McConway, K.J. and Ostrowski, E (1994) *A Handbook of Small Data Sets*. Chapman and Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Ellis, M.E., Neal, K.R. and Webb, A.K. (1987) Is smoking a risk factor for pneumonia in patients with chickenpox? *British Medical Journal*, **294**, 1002.

coal *Dates of Coal Mining Disasters*

Description

The `coal` data frame has 191 rows and 1 columns.

This data frame gives the dates of 191 explosions in coal mines which resulted in 10 or more fatalities. The time span of the data is from March 15, 1851 until March 22 1962.

Usage

```
coal
```

Format

This data frame contains the following column:

`date` The date of the disaster. The integer part of `date` gives the year. The day is represented as the fraction of the year that had elapsed on that day.

Source

The data were obtained from

Hand, D.J., Daly, F., Lunn, A.D., McConway, K.J. and Ostrowski, E. (1994) *A Handbook of Small Data Sets*, Chapman and Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Jarrett, R.G. (1979) A note on the intervals between coal-mining disasters. *Biometrika*, **66**, 191-193.

control *Control Variate Calculations*

Description

This function will find control variate estimates from a bootstrap output object. It can either find the adjusted bias estimate using post-simulation balancing or it can estimate the bias, variance, third cumulant and quantiles, using the linear approximation as a control variate.

Usage

```
control(boot.out, L = NULL, distn = NULL, index = 1, t0 = NULL,
        t = NULL, bias.adj = FALSE, alpha = NULL, ...)
```

Arguments

boot.out	A bootstrap output object returned from boot. The bootstrap replicates must have been generated using the usual nonparametric bootstrap.
L	The empirical influence values for the statistic of interest. If L is not supplied then empinf is called to calculate them from boot.out.
distn	If present this must be the output from smooth.spline giving the distribution function of the linear approximation. This is used only if bias.adj is FALSE. Normally this would be found using a saddlepoint approximation. If it is not supplied in that case then it is calculated by saddle.distn.
index	The index of the variable of interest in the output of boot.out$statistic.
t0	The observed value of the statistic of interest on the original data set boot.out$data. This argument is used only if bias.adj is FALSE. The input value is ignored if t is not also supplied. The default value is is boot.out$t0[index].
t	The bootstrap replicate values of the statistic of interest. This argument is used only if bias.adj is FALSE. The input is ignored if t0 is not supplied also. The default value is boot.out$t[,index].
bias.adj	A logical variable which if TRUE specifies that the adjusted bias estimate using post-simulation balance is all that is required. If bias.adj is FALSE (default) then the linear approximation to the statistic is calculated and used as a control variate in estimates of the bias, variance and third cumulant as well as quantiles.
alpha	The alpha levels for the required quantiles if bias.adj is FALSE.
...	Any additional arguments that boot.out$statistic requires. These are passed unchanged every time boot.out$statistic is called. boot.out$statistic is called once if bias.adj is TRUE, otherwise it may be called by empinf for empirical influence calculations if L is not supplied.

Details

If bias.adj is FALSE then the linear approximation to the statistic is found and evaluated at each bootstrap replicate. Then using the equation $T^* = Tl^* + (T^* - Tl^*)$, moment estimates can be found. For quantile estimation the distribution of the linear approximation to t is approximated very accurately by saddlepoint methods, this is then combined with the bootstrap replicates to approximate the bootstrap distribution of t and hence to estimate the bootstrap quantiles of t.

Value

If bias.adj is TRUE then the returned value is the adjusted bias estimate.

If bias.adj is FALSE then the returned value is a list with the following components

L	The empirical influence values used. These are the input values if supplied, and otherwise they are the values calculated by empinf.
tL	The linear approximations to the bootstrap replicates t of the statistic of interest.
bias	The control estimate of bias using the linear approximation to t as a control variate.

var	The control estimate of variance using the linear approximation to t as a control variate.
k3	The control estimate of the third cumulant using the linear approximation to t as a control variate.
quantiles	A matrix with two columns; the first column are the alpha levels used for the quantiles and the second column gives the corresponding control estimates of the quantiles using the linear approximation to t as a control variate.
distn	An output object from smooth.spline describing the saddlepoint approximation to the bootstrap distribution of the linear approximation to t. If distn was supplied on input then this is the same as the input otherwise it is calculated by a call to saddle.distn.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Davison, A.C., Hinkley, D.V. and Schechtman, E. (1986) Efficient bootstrap simulation. *Biometrika*, **73**, 555–566.

Efron, B. (1990) More efficient bootstrap computations. *Journal of the American Statistical Association*, **55**, 79–89.

See Also

boot, empinf, k3.linear, linear.approx, saddle.distn, smooth.spline, var.linear

Examples

```
# Use of control variates for the variance of the air-conditioning data
mean.fun <- function(d, i)
{    m <- mean(d$hours[i])
     n <- nrow(d)
     v <- (n-1)*var(d$hours[i])/n^2
     c(m, v)
}
air.boot <- boot(aircondit, mean.fun, R = 999)
control(air.boot, index = 2, bias.adj = TRUE)
air.cont <- control(air.boot, index = 2)
# Now let us try the variance on the log scale.
air.cont1 <- control(air.boot, t0 = log(air.boot$t0[2]),
                     t = log(air.boot$t[, 2]))
```

---

corr *Correlation Coefficient*

Description

Calculates the weighted correlation given a data set and a set of weights.

Usage

```
corr(d, w = rep(1, nrow(d))/nrow(d))
```

Arguments

d	A matrix with two columns corresponding to the two variables whose correlation we wish to calculate.
w	A vector of weights to be applied to each pair of observations. The default is equal weights for each pair. Normalization takes place within the function so `sum(w)` need not equal 1.

Details

This function finds the correlation coefficient in weighted form. This is often useful in bootstrap methods since it allows for numerical differentiation to get the empirical influence values. It is also necessary to have the statistic in this form to find ABC intervals.

Value

The correlation coefficient between `d[,1]` and `d[,2]`.

See Also

cor

---

cum3 *Calculate Third Order Cumulants*

Description

Calculates an estimate of the third cumulant, or skewness, of a vector. Also, if more than one vector is specified, a product-moment of order 3 is estimated.

Usage

```
cum3(a, b = a, c = a, unbiased = TRUE)
```

Arguments

a	A vector of observations.
b	Another vector of observations, if not supplied it is set to the value of `a`. If supplied then it must be the same length as `a`.
c	Another vector of observations, if not supplied it is set to the value of `a`. If supplied then it must be the same length as `a`.
unbiased	A logical value indicating whether the unbiased estimator should be used.

Details

The unbiased estimator uses a multiplier of `n/((n-1)*(n-2))` where `n` is the sample size, if `unbiased` is `FALSE` then a multiplier of `1/n` is used. This is multiplied by `sum((a-mean(a))*(b-mean(b))*(c-mean(c)))` to give the required estimate.

Value

The required estimate.

---

cv.glm *Cross-validation for Generalized Linear Models*

Description

This function calculates the estimated K-fold cross-validation prediction error for generalized linear models.

Usage

```
cv.glm(data, glmfit, cost, K)
```

Arguments

`data`	A matrix or data frame containing the data. The rows should be cases and the columns correspond to variables, one of which is the response.
`glmfit`	An object of class `"glm"` containing the results of a generalized linear model fitted to `data`.
`cost`	A function of two vector arguments specifying the cost function for the cross-validation. The first argument to `cost` should correspond to the observed responses and the second argument should correspond to the predicted or fitted responses from the generalized linear model. `cost` must return a non-negative scalar value. The default is the average squared error function.
`K`	The number of groups into which the data should be split to estimate the cross-validation prediction error. The value of `K` must be such that all groups are of approximately equal size. If the supplied value of `K` does not satisfy this criterion then it will be set to the closest integer which does and a warning is generated specifying the value of `K` used. The default is to set `K` equal to the number of observations in `data` which gives the usual leave-one-out cross-validation.

Details

The data is divided randomly into `K` groups. For each group the generalized linear model is fit to `data` omitting that group, then the function `cost` is applied to the observed responses in the group that was omitted from the fit and the prediction made by the fitted models for those observations.

When `K` is the number of observations leave-one-out cross-validation is used and all the possible splits of the data are used. When `K` is less than the number of observations the `K` splits to be used

are found by randomly partitioning the data into K groups of approximately equal size. In this latter case a certain amount of bias is introduced. This can be reduced by using a simple adjustment (see equation 6.48 in Davison and Hinkley, 1997). The second value returned in delta is the estimate adjusted by this method.

Value

The returned value is a list with the following components.

call	The original call to cv.glm.
K	The value of K used for the K-fold cross validation.
delta	A vector of length two. The first component is the raw cross-validation estimate of prediction error. The second component is the adjusted cross-validation estimate. The adjustment is designed to compensate for the bias introduced by not using leave-one-out cross-validation.
seed	The value of .Random.seed when cv.glm was called.

Side Effects

The value of .Random.seed is updated.

References

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classification and Regression Trees*. Wadsworth.

Burman, P. (1989) A comparative study of ordinary cross-validation, *v*-fold cross-validation and repeated learning-testing methods. *Biometrika*, **76**, 503–514

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1986) How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association*, **81**, 461–470.

Stone, M. (1974) Cross-validation choice and assessment of statistical predictions (with Discussion). *Journal of the Royal Statistical Society, B*, **36**, 111–147.

See Also

glm, glm.diag, predict

Examples

```
# leave-one-out and 6-fold cross-validation prediction error for
# the mammals data set.
data(mammals, package="MASS")
mammals.glm <- glm(log(brain) ~ log(body), data = mammals)
(cv.err <- cv.glm(mammals, mammals.glm)$delta)
(cv.err.6 <- cv.glm(mammals, mammals.glm, K = 6)$delta)

# As this is a linear model we could calculate the leave-one-out
```

```
# cross-validation estimate without any extra model-fitting.
muhat <- fitted(mammals.glm)
mammals.diag <- glm.diag(mammals.glm)
(cv.err <- mean((mammals.glm$y - muhat)^2/(1 - mammals.diag$h)^2))


# leave-one-out and 11-fold cross-validation prediction error for
# the nodal data set.  Since the response is a binary variable an
# appropriate cost function is
cost <- function(r, pi = 0) mean(abs(r-pi) > 0.5)

nodal.glm <- glm(r ~ stage+xray+acid, binomial, data = nodal)
(cv.err <- cv.glm(nodal, nodal.glm, cost, K = nrow(nodal))$delta)
(cv.11.err <- cv.glm(nodal, nodal.glm, cost, K = 11)$delta)
```

`darwin` *Darwin's Plant Height Differences*

Description

The `darwin` data frame has 15 rows and 1 columns.

Charles Darwin conducted an experiment to examine the superiority of cross-fertilized plants over self-fertilized plants. 15 pairs of plants were used. Each pair consisted of one cross-fertilized plant and one self-fertilized plant which germinated at the same time and grew in the same pot. The plants were measured at a fixed time after planting and the difference in heights between the cross- and self-fertilized plants are recorded in eighths of an inch.

Usage

```
darwin
```

Format

This data frame contains the following column:

`y` The difference in heights for the pairs of plants (in units of 0.125 inches).

Source

The data were obtained from

Fisher, R.A. (1935) *Design of Experiments*. Oliver and Boyd.

References

Darwin, C. (1876) *The Effects of Cross- and Self-fertilisation in the Vegetable Kingdom*. John Murray.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

dogs *Cardiac Data for Domestic Dogs*

Description

The dogs data frame has 7 rows and 2 columns.

Data on the cardiac oxygen consumption and left ventricular pressure were gathered on 7 domestic dogs.

Usage

```
dogs
```

Format

This data frame contains the following columns:

mvo Cardiac Oxygen Consumption

lvp Left Ventricular Pressure

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

downs.bc *Incidence of Down's Syndrome in British Columbia*

Description

The downs.bc data frame has 30 rows and 3 columns.

Down's syndrome is a genetic disorder caused by an extra chromosome 21 or a part of chromosome 21 being translocated to another chromosome. The incidence of Down's syndrome is highly dependent on the mother's age and rises sharply after age 30. In the 1960's a large scale study of the effect of maternal age on the incidence of Down's syndrome was conducted at the British Columbia Health Surveillance Registry. These are the data which was collected in that study.

Mothers were classified by age. Most groups correspond to the age in years but the first group comprises all mothers with ages in the range 15-17 and the last is those with ages 46-49. No data for mothers over 50 or below 15 were collected.

Usage

```
downs.bc
```

Format

This data frame contains the following columns:

`age` The average age of all mothers in the age category.

`m` The total number of live births to mothers in the age category.

`r` The number of cases of Down's syndrome.

Source

The data were obtained from

Geyer, C.J. (1991) Constrained maximum likelihood exemplified by isotonic convex logistic regression. *Journal of the American Statistical Association*, **86**, 717–724.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`ducks` *Behavioral and Plumage Characteristics of Hybrid Ducks*

Description

The `ducks` data frame has 11 rows and 2 columns.

Each row of the data frame represents a male duck who is a second generation cross of mallard and pintail ducks. For 11 such ducks a behavioural and plumage index were calculated. These were measured on scales devised for this experiment which was to examine whether there was any link between which species the ducks resembled physically and which they resembled in behaviour. The scale for the physical appearance ranged from 0 (identical in appearance to a mallard) to 20 (identical to a pintail). The behavioural traits of the ducks were on a scale from 0 to 15 with lower numbers indicating closer to mallard-like in behaviour.

Usage

```
ducks
```

Format

This data frame contains the following columns:

`plumage` The index of physical appearance based on the plumage of individual ducks.

`behaviour` The index of behavioural characteristics of the ducks.

Source

The data were obtained from

Larsen, R.J. and Marx, M.L. (1986) *An Introduction to Mathematical Statistics and its Applications* (Second Edition). Prentice-Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Sharpe, R.S., and Johnsgard, P.A. (1966) Inheritance of behavioral characters in F_2 mallard x pintail (*Anas Platyrhynchos L. x Anas Acuta L.*) hybrids. *Behaviour*, **27**, 259-272.

EEF.profile *Empirical Likelihoods*

Description

Construct the empirical log likelihood or empirical exponential family log likelihood for a mean.

Usage

```
EEF.profile(y, tmin = min(y) + 0.1, tmax = max(y) - 0.1, n.t = 25,
            u = function(y, t) y - t)
EL.profile(y, tmin  =  min(y) + 0.1, tmax = max(y) - 0.1, n.t  =  25,
           u = function(y, t) y - t)
```

Arguments

`y`	A vector or matrix of data
`tmin`	The minimum value of the range over which the likelihood should be computed. This must be larger than `min(y)`.
`tmax`	The maximum value of the range over which the likelihood should be computed. This must be smaller than `max(y)`.
`n.t`	The number of points between `tmin` and `tmax` at which the value of the log-likelihood should be computed.
`u`	A function of the data and the parameter.

Details

These functions calculate the log likelihood for a mean using either an empirical likelihood or an empirical exponential family likelihood. They are supplied as part of the package `boot` for demonstration purposes with the practicals in chapter 10 of Davison and Hinkley (1997). The functions are not intended for general use and are not supported as part of the `boot`package. For more general and more robust code to calculate empirical likelihoods see Professor A. B. Owen's empirical likelihood home page at the URL http://statistics.stanford.edu/~owen/empirical/.

Value

A matrix with `n.t` rows. The first column contains the values of the parameter used. The second column of the output of `EL.profile` contains the values of the empirical log likelihood. The second and third columns of the output of `EEF.profile` contain two versions of the empirical exponential family log-likelihood. The final column of the output matrix contains the values of the Lagrange multiplier used in the optimization procedure.

Author(s)

Angelo J. Canty

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`empinf` *Empirical Influence Values*

Description

This function calculates the empirical influence values for a statistic applied to a data set. It allows four types of calculation, namely the infinitesimal jackknife (using numerical differentiation), the usual jackknife estimates, the 'positive' jackknife estimates and a method which estimates the empirical influence values using regression of bootstrap replicates of the statistic. All methods can be used with one or more samples.

Usage

```
empinf(boot.out = NULL, data = NULL, statistic = NULL,
       type = NULL, stype = NULL ,index = 1, t = NULL,
       strata = rep(1, n), eps = 0.001, ...)
```

Arguments

`boot.out`	A bootstrap object created by the function `boot`. If `type` is `"reg"` then this argument is required. For any of the other types it is an optional argument. If it is included when optional then the values of `data`, `statistic`, `stype`, and `strata` are taken from the components of `boot.out` and any values passed to `empinf` directly are ignored.
`data`	A vector, matrix or data frame containing the data for which empirical influence values are required. It is a required argument if `boot.out` is not supplied. If `boot.out` is supplied then `data` is set to `boot.out$data` and any value supplied is ignored.

`statistic` The statistic for which empirical influence values are required. It must be a function of at least two arguments, the data set and a vector of weights, frequencies or indices. The nature of the second argument is given by the value of `stype`. Any other arguments that it takes must be supplied to `empinf` and will be passed to `statistic` unchanged. This is a required argument if `boot.out` is not supplied, otherwise its value is taken from `boot.out` and any value supplied here will be ignored.

`type` The calculation type to be used for the empirical influence values. Possible values of `type` are `"inf"` (infinitesimal jackknife), `"jack"` (usual jackknife), `"pos"` (positive jackknife), and `"reg"` (regression estimation). The default value depends on the other arguments. If `t` is supplied then the default value of `type` is `"reg"` and `boot.out` should be present so that its frequency array can be found. It `t` is not supplied then if `stype` is `"w"`, the default value of `type` is `"inf"`; otherwise, if `boot.out` is present the default is `"reg"`. If none of these conditions apply then the default is `"jack"`. Note that it is an error for `type` to be `"reg"` if `boot.out` is missing or to be `"inf"` if `stype` is not `"w"`.

`stype` A character variable giving the nature of the second argument to `statistic`. It can take on three values: `"w"` (weights), `"f"` (frequencies), or `"i"` (indices). If `boot.out` is supplied the value of `stype` is set to `boot.out$stype` and any value supplied here is ignored. Otherwise it is an optional argument which defaults to `"w"`. If `type` is `"inf"` then `stype` MUST be `"w"`.

`index` An integer giving the position of the variable of interest in the output of `statistic`.

`t` A vector of length `boot.out$R` which gives the bootstrap replicates of the statistic of interest. `t` is used only when `type` is `reg` and it defaults to `boot.out$t[,index]`.

`strata` An integer vector or a factor specifying the strata for multi-sample problems. If `boot.out` is supplied the value of `strata` is set to `boot.out$strata`. Otherwise it is an optional argument which has default corresponding to the single sample situation.

`eps` This argument is used only if `type` is `"inf"`. In that case the value of epsilon to be used for numerical differentiation will be `eps` divided by the number of observations in `data`.

`...` Any other arguments that `statistic` takes. They will be passed unchanged to `statistic` every time that it is called.

Details

If `type` is `"inf"` then numerical differentiation is used to approximate the empirical influence values. This makes sense only for statistics which are written in weighted form (i.e. `stype` is `"w"`). If `type` is `"jack"` then the usual leave-one-out jackknife estimates of the empirical influence are returned. If `type` is `"pos"` then the positive (include-one-twice) jackknife values are used. If `type` is `"reg"` then a bootstrap object must be supplied. The regression method then works by regressing the bootstrap replicates of `statistic` on the frequency array from which they were derived. The bootstrap frequency array is obtained through a call to `boot.array`. Further details of the methods are given in Section 2.7 of Davison and Hinkley (1997).

Empirical influence values are often used frequently in nonparametric bootstrap applications. For this reason many other functions call `empinf` when they are required. Some examples of their use

are for nonparametric delta estimates of variance, BCa intervals and finding linear approximations to statistics for use as control variates. They are also used for antithetic bootstrap resampling.

Value

A vector of the empirical influence values of `statistic` applied to `data`. The values will be in the same order as the observations in data.

Warning

All arguments to `empinf` must be passed using the `name = value` convention. If this is not followed then unpredictable errors can occur.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1982) *The Jackknife, the Bootstrap and Other Resampling Plans*. CBMS-NSF Regional Conference Series in Applied Mathematics, **38**, SIAM.

Fernholtz, L.T. (1983) *von Mises Calculus for Statistical Functionals*. Lecture Notes in Statistics, **19**, Springer-Verlag.

See Also

`boot`, `boot.array`, `boot.ci`, `control`, `jack.after.boot`, `linear.approx`, `var.linear`

Examples

```
# The empirical influence values for the ratio of means in
# the city data.
ratio <- function(d, w) sum(d$x *w)/sum(d$u*w)
empinf(data = city, statistic = ratio)
city.boot <- boot(city, ratio, 499, stype="w")
empinf(boot.out = city.boot, type = "reg")

# A statistic that may be of interest in the difference of means
# problem is the t-statistic for testing equality of means.  In
# the bootstrap we get replicates of the difference of means and
# the variance of that statistic and then want to use this output
# to get the empirical influence values of the t-statistic.
grav1 <- gravity[as.numeric(gravity[,2]) >= 7,]
grav.fun <- function(dat, w) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2] - mns[1], s2hat)
}
```

```
grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w",
                  strata = grav1[, 2])

# Since the statistic of interest is a function of the bootstrap
# statistics, we must calculate the bootstrap replicates and pass
# them to empinf using the t argument.
grav.z <- (grav.boot$t[,1]-grav.boot$t0[1])/sqrt(grav.boot$t[,2])
empinf(boot.out = grav.boot, t = grav.z)
```

`envelope` *Confidence Envelopes for Curves*

Description

This function calculates overall and pointwise confidence envelopes for a curve based on bootstrap replicates of the curve evaluated at a number of fixed points.

Usage

```
envelope(boot.out = NULL, mat = NULL, level = 0.95, index = 1:ncol(mat))
```

Arguments

`boot.out`	An object of class `"boot"` for which `boot.out$t` contains the replicates of the curve at a number of fixed points.
`mat`	A matrix of bootstrap replicates of the values of the curve at a number of fixed points. This is a required argument if `boot.out` is not supplied and is set to `boot.out$t` otherwise.
`level`	The confidence level of the envelopes required. The default is to find 95% confidence envelopes. It can be a scalar or a vector of length 2. If it is scalar then both the pointwise and the overall envelopes are found at that level. If is a vector then the first element gives the level for the pointwise envelope and the second gives the level for the overall envelope.
`index`	The numbers of the columns of `mat` which contain the bootstrap replicates. This can be used to ensure that other statistics which may have been calculated in the bootstrap are not considered as values of the function.

Details

The pointwise envelope is found by simply looking at the quantiles of the replicates at each point. The overall error for that envelope is then calculated using equation (4.17) of Davison and Hinkley (1997). A sequence of pointwise envelopes is then found until one of them has overall error approximately equal to the level required. If no such envelope can be found then the envelope returned will just contain the extreme values of each column of `mat`.

Value

A list with the following components :

`point`	A matrix with two rows corresponding to the values of the upper and lower pointwise confidence envelope at the same points as the bootstrap replicates were calculated.
`overall`	A matrix similar to `point` but containing the envelope which controls the overall error.
`k.pt`	The quantiles used for the pointwise envelope.
`err.pt`	A vector with two components, the first gives the pointwise error rate for the pointwise envelope, and the second the overall error rate for that envelope.
`k.ov`	The quantiles used for the overall envelope.
`err.ov`	A vector with two components, the first gives the pointwise error rate for the overall envelope, and the second the overall error rate for that envelope.
`err.nom`	A vector of length 2 giving the nominal error rates for the pointwise and the overall envelopes.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

`boot`, `boot.ci`

Examples

```
# Testing whether the final series of measurements of the gravity data
# may come from a normal distribution.  This is done in Examples 4.7
# and 4.8 of Davison and Hinkley (1997).
grav1 <- gravity$g[gravity$series == 8]
grav.z <- (grav1 - mean(grav1))/sqrt(var(grav1))
grav.gen <- function(dat, mle) rnorm(length(dat))
grav.qqboot <- boot(grav.z, sort, R = 999, sim = "parametric",
                    ran.gen = grav.gen)
grav.qq <- qqnorm(grav.z, plot.it = FALSE)
grav.qq <- lapply(grav.qq, sort)
plot(grav.qq, ylim = c(-3.5, 3.5), ylab = "Studentized Order Statistics",
     xlab = "Normal Quantiles")
grav.env <- envelope(grav.qqboot, level = 0.9)
lines(grav.qq$x, grav.env$point[1, ], lty = 4)
lines(grav.qq$x, grav.env$point[2, ], lty = 4)
lines(grav.qq$x, grav.env$overall[1, ], lty = 1)
lines(grav.qq$x, grav.env$overall[2, ], lty = 1)
```

`exp.tilt` *Exponential Tilting*

Description

This function calculates exponentially tilted multinomial distributions such that the resampling distributions of the linear approximation to a statistic have the required means.

Usage

```
exp.tilt(L, theta = NULL, t0 = 0, lambda = NULL,
         strata = rep(1, length(L)))
```

Arguments

`L`	The empirical influence values for the statistic of interest based on the observed data. The length of `L` should be the same as the size of the original data set. Typically `L` will be calculated by a call to `empinf`.
`theta`	The value at which the tilted distribution is to be centred. This is not required if `lambda` is supplied but is needed otherwise.
`t0`	The current value of the statistic. The default is that the statistic equals 0.
`lambda`	The Lagrange multiplier(s). For each value of `lambda` a multinomial distribution is found with probabilities proportional to `exp(lambda * L)`. In general `lambda` is not known and so `theta` would be supplied, and the corresponding value of `lambda` found. If both `lambda` and `theta` are supplied then `lambda` is ignored and the multipliers for tilting to `theta` are found.
`strata`	A vector or factor of the same length as `L` giving the strata for the observed data and the empirical influence values `L`.

Details

Exponential tilting involves finding a set of weights for a data set to ensure that the bootstrap distribution of the linear approximation to a statistic of interest has mean `theta`. The weights chosen to achieve this are given by `p[j]` proportional to `exp(lambda*L[j]/n)`, where `n` is the number of data points. `lambda` is then chosen to make the mean of the bootstrap distribution, of the linear approximation to the statistic of interest, equal to the required value `theta`. Thus `lambda` is defined as the solution of a nonlinear equation. The equation is solved by minimizing the Euclidean distance between the left and right hand sides of the equation using the function `nlmin`. If this minimum is not equal to zero then the method fails.

Typically exponential tilting is used to find suitable weights for importance resampling. If a small tail probability or quantile of the distribution of the statistic of interest is required then a more efficient simulation is to centre the resampling distribution close to the point of interest and then use the functions `imp.prob` or `imp.quantile` to estimate the required quantity.

Another method of achieving a similar shifting of the distribution is through the use of `smooth.f`. The function `tilt.boot` uses `exp.tilt` or `smooth.f` to find the weights for a tilted bootstrap.

Value

A list with the following components :

p	The tilted probabilities. There will be m distributions where m is the length of theta (or lambda if supplied). If m is 1 then p is a vector of length(L) probabilities. If m is greater than 1 then p is a matrix with m rows, each of which contain length(L) probabilities. In this case the vector p[i,] is the distribution tilted to theta[i]. p is in the form required by the argument weights of the function boot for importance resampling.
lambda	The Lagrange multiplier used in the equation to determine the tilted probabilities. lambda is a vector of the same length as theta.
theta	The values of theta to which the distributions have been tilted. In general this will be the input value of theta but if lambda was supplied then this is the vector of the corresponding theta values.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1981) Nonparametric standard errors and confidence intervals (with Discussion). *Canadian Journal of Statistics*, **9**, 139–172.

See Also

empinf, imp.prob, imp.quantile, optim, smooth.f, tilt.boot

Examples

```
# Example 9.8 of Davison and Hinkley (1997) requires tilting the resampling
# distribution of the studentized statistic to be centred at the observed
# value of the test statistic 1.84.  This can be achieved as follows.
grav1 <- gravity[as.numeric(gravity[,2]) >=7 , ]
grav.fun <- function(dat, w, orig) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2]-mns[1], s2hat, (mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.L <- empinf(data = grav1, statistic = grav.fun, stype = "w",
                 strata = grav1[,2], index = 3, orig = grav.z0[1])
grav.tilt <- exp.tilt(grav.L, grav.z0[3], strata = grav1[,2])
boot(grav1, grav.fun, R = 499, stype = "w", weights = grav.tilt$p,
     strata = grav1[,2], orig = grav.z0[1])
```

fir *Counts of Balsam-fir Seedlings*

Description

The `fir` data frame has 50 rows and 3 columns.

The number of balsam-fir seedlings in each quadrant of a grid of 50 five foot square quadrants were counted. The grid consisted of 5 rows of 10 quadrants in each row.

Usage

```
fir
```

Format

This data frame contains the following columns:

`count` The number of seedlings in the quadrant.

`row` The row number of the quadrant.

`col` The quadrant number within the row.

Source

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

freq.array *Bootstrap Frequency Arrays*

Description

Take a matrix of indices for nonparametric bootstrap resamples and return the frequencies of the original observations in each resample.

Usage

```
freq.array(i.array)
```

Arguments

`i.array`	This will be an matrix of integers between 1 and n, where n is the number of observations in a data set. The matrix will have n columns and R rows where R is the number of bootstrap resamples. Such matrices are found by `boot` when doing nonparametric bootstraps. They can also be found after a bootstrap has been run through the function `boot.array`.

Value

A matrix of the same dimensions as the input matrix. Each row of the matrix corresponds to a single bootstrap resample. Each column of the matrix corresponds to one of the original observations and specifies its frequency in each bootstrap resample. Thus the first column tells us how often the first observation appeared in each bootstrap resample. Such frequency arrays are often useful for diagnostic purposes such as the jackknife-after-bootstrap plot. They are also necessary for the regression estimates of empirical influence values and for finding importance sampling weights.

See Also

boot.array

---

frets *Head Dimensions in Brothers*

Description

The frets data frame has 25 rows and 4 columns.

The data consist of measurements of the length and breadth of the heads of pairs of adult brothers in 25 randomly sampled families. All measurements are expressed in millimetres.

Usage

```
frets
```

Format

This data frame contains the following columns:

l1 The head length of the eldest son.

b1 The head breadth of the eldest son.

l2 The head length of the second son.

b2 The head breadth of the second son.

Source

The data were obtained from

Frets, G.P. (1921) Heredity of head form in man. *Genetica*, **3**, 193.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Whittaker, J. (1990) *Graphical Models in Applied Multivariate Statistics*. John Wiley.

`glm.diag` *Generalized Linear Model Diagnostics*

Description

Calculates jackknife deviance residuals, standardized deviance residuals, standardized Pearson residuals, approximate Cook statistic, leverage and estimated dispersion.

Usage

```
glm.diag(glmfit)
```

Arguments

`glmfit`	`glmfit` is a `glm.object` - the result of a call to `glm()`

Value

Returns a list with the following components

`res`	The vector of jackknife deviance residuals.
`rd`	The vector of standardized deviance residuals.
`rp`	The vector of standardized Pearson residuals.
`cook`	The vector of approximate Cook statistics.
`h`	The vector of leverages of the observations.
`sd`	The value used to standardize the residuals. This is the estimate of residual standard deviation in the Gaussian family and is the square root of the estimated shape parameter in the Gamma family. In all other cases it is 1.

Note

See the help for `glm.diag.plots` for an example of the use of `glm.diag`.

References

Davison, A.C. and Snell, E.J. (1991) Residuals and diagnostics. In *Statistical Theory and Modelling: In Honour of Sir David Cox*. D.V. Hinkley, N. Reid and E.J. Snell (editors), 83–106. Chapman and Hall.

See Also

`glm`, `glm.diag.plots`, `summary.glm`

`glm.diag.plots` *Diagnostics plots for generalized linear models*

Description

Makes plot of jackknife deviance residuals against linear predictor, normal scores plots of standardized deviance residuals, plot of approximate Cook statistics against leverage/(1-leverage), and case plot of Cook statistic.

Usage

```
glm.diag.plots(glmfit, glmdiag = glm.diag(glmfit), subset = NULL,
               iden = FALSE, labels = NULL, ret = FALSE)
```

Arguments

`glmfit`	`glm.object` : the result of a call to `glm()`
`glmdiag`	Diagnostics of `glmfit` obtained from a call to `glm.diag`. If it is not supplied then it is calculated.
`subset`	Subset of `data` for which `glm` fitting performed: should be the same as the `subset` option used in the call to `glm()` which generated `glmfit`. Needed only if the `subset=` option was used in the call to `glm`.
`iden`	A logical argument. If `TRUE` then, after the plots are drawn, the user will be prompted for an integer between 0 and 4. A positive integer will select a plot and invoke `identify()` on that plot. After exiting `identify()`, the user is again prompted, this loop continuing until the user responds to the prompt with 0. If `iden` is `FALSE` (default) the user cannot interact with the plots.
`labels`	A vector of labels for use with `identify()` if `iden` is `TRUE`. If it is not supplied then the labels are derived from `glmfit`.
`ret`	A logical argument indicating if `glmdiag` should be returned. The default is `FALSE`.

Details

The diagnostics required for the plots are calculated by `glm.diag`. These are then used to produce the four plots on the current graphics device.

The plot on the top left is a plot of the jackknife deviance residuals against the fitted values.

The plot on the top right is a normal QQ plot of the standardized deviance residuals. The dotted line is the expected line if the standardized residuals are normally distributed, i.e. it is the line with intercept 0 and slope 1.

The bottom two panels are plots of the Cook statistics. On the left is a plot of the Cook statistics against the standardized leverages. In general there will be two dotted lines on this plot. The horizontal line is at 8/(n-2p) where n is the number of observations and p is the number of parameters estimated. Points above this line may be points with high influence on the model. The vertical line is at 2p/(n-2p) and points to the right of this line have high leverage compared to the variance of the

raw residual at that point. If all points are below the horizontal line or to the left of the vertical line then the line is not shown.

The final plot again shows the Cook statistic this time plotted against case number enabling us to find which observations are influential.

Use of `iden=T` is encouraged for proper exploration of these four plots as a guide to how well the model fits the data and whether certain observations have an unduly large effect on parameter estimates.

Value

If `ret` is `TRUE` then the value of `glmdiag` is returned otherwise there is no returned value.

Side Effects

The current device is cleared and four plots are plotted by use of `split.screen(c(2,2))`. If `iden` is `TRUE`, interactive identification of points is enabled. All screens are closed, but not cleared, on termination of the function.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Davison, A.C. and Snell, E.J. (1991) Residuals and diagnostics. In *Statistical Theory and Modelling: In Honour of Sir David Cox* D.V. Hinkley, N. Reid, and E.J. Snell (editors), 83–106. Chapman and Hall.

See Also

`glm`, `glm.diag`, `identify`

Examples

```
# In this example we look at the leukaemia data which was looked at in
# Example 7.1 of Davison and Hinkley (1997)
data(leuk, package = "MASS")
leuk.mod <- glm(time ~ ag-1+log10(wbc), family = Gamma(log), data = leuk)
leuk.diag <- glm.diag(leuk.mod)
glm.diag.plots(leuk.mod, leuk.diag)
```

`gravity` *Acceleration Due to Gravity*

Description

The `gravity` data frame has 81 rows and 2 columns.

The `grav` data set has 26 rows and 2 columns.

Between May 1934 and July 1935, the National Bureau of Standards in Washington D.C. conducted a series of experiments to estimate the acceleration due to gravity, g, at Washington. Each experiment produced a number of replicate estimates of g using the same methodology. Although the basic method remained the same for all experiments, that of the reversible pendulum, there were changes in configuration.

The `gravity` data frame contains the data from all eight experiments. The `grav` data frame contains the data from the experiments 7 and 8. The data are expressed as deviations from 980.000 in centimetres per second squared.

Usage

```
gravity
```

Format

This data frame contains the following columns:

`g` The deviation of the estimate from 980.000 centimetres per second squared.

`series` A factor describing from which experiment the estimate was derived.

Source

The data were obtained from

Cressie, N. (1982) Playing safe with misweighted means. *Journal of the American Statistical Association*, **77**, 754–759.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`hirose` *Failure Time of PET Film*

Description

The `hirose` data frame has 44 rows and 3 columns.

PET film is used in electrical insulation. In this accelerated life test the failure times for 44 samples in gas insulated transformers. 4 different voltage levels were used.

Usage

```
hirose
```

Format

This data frame contains the following columns:

`volt` The voltage (in kV).

`time` The failure or censoring time in hours.

`cens` The censoring indicator; 1 means right-censored data.

Source

The data were obtained from

Hirose, H. (1993) Estimation of threshold stress in accelerated life-testing. *IEEE Transactions on Reliability*, **42**, 650–657.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`Imp.Estimates` *Importance Sampling Estimates*

Description

Central moment, tail probability, and quantile estimates for a statistic under importance resampling.

Usage

```
imp.moments(boot.out = NULL, index = 1, t = boot.out$t[, index],
            w = NULL, def = TRUE, q = NULL)
imp.prob(boot.out = NULL, index = 1, t0 = boot.out$t0[index],
         t = boot.out$t[, index], w = NULL, def = TRUE, q = NULL)
imp.quantile(boot.out = NULL, alpha = NULL, index = 1,
             t = boot.out$t[, index], w = NULL, def = TRUE, q = NULL)
```

Arguments

`boot.out`	A object of class `"boot"` generated by a call to `boot` or `tilt.boot`. Use of these functions makes sense only when the bootstrap resampling used unequal weights for the observations. If the importance weights `w` are not supplied then `boot.out` is a required argument. It is also required if `t` is not supplied.
`alpha`	The alpha levels for the required quantiles. The default is to calculate the 1%, 2.5%, 5%, 10%, 90%, 95%, 97.5% and 99% quantiles.
`index`	The index of the variable of interest in the output of `boot.out$statistic`. This is not used if the argument `t` is supplied.

`t0` The values at which tail probability estimates are required. For each value `t0[i]` the function will estimate the bootstrap cdf evaluated at `t0[i]`. If `imp.prob` is called without the argument `t0` then the bootstrap cdf evaluated at the observed value of the statistic is found.

`t` The bootstrap replicates of a statistic. By default these are taken from the bootstrap output object `boot.out` but they can be supplied separately if required (e.g. when the statistic of interest is a function of the calculated values in `boot.out`). Either `boot.out` or `t` must be supplied.

`w` The importance resampling weights for the bootstrap replicates. If they are not supplied then `boot.out` must be supplied, in which case the importance weights are calculated by a call to `imp.weights`.

`def` A logical value indicating whether a defensive mixture is to be used for weight calculation. This is used only if `w` is missing and it is passed unchanged to `imp.weights` to calculate `w`.

`q` A vector of probabilities specifying the resampling distribution from which any estimates should be found. In general this would correspond to the usual bootstrap resampling distribution which gives equal weight to each of the original observations. The estimates depend on this distribution only through the importance weights `w` so this argument is ignored if `w` is supplied. If `w` is missing then `q` is passed as an argument to `imp.weights` and used to find `w`.

Value

A list with the following components :

`alpha` The `alpha` levels used for the quantiles, if `imp.quantile` is used.

`t0` The values at which the tail probabilities are estimated, if `imp.prob` is used.

`raw` The raw importance resampling estimates. For `imp.moments` this has length 2, the first component being the estimate of the mean and the second being the variance estimate. For `imp.prob`, `raw` is of the same length as `t0`, and for `imp.quantile` it is of the same length as `alpha`.

`rat` The ratio importance resampling estimates. In this method the weights `w` are rescaled to have average value one before they are used. The format of this vector is the same as `raw`.

`reg` The regression importance resampling estimates. In this method the weights which are used are derived from a regression of `t*w` on `w`. This choice of weights can be shown to minimize the variance of the weights and also the Euclidean distance of the weights from the uniform weights. The format of this vector is the same as `raw`.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Hesterberg, T. (1995) Weighted average importance sampling and defensive mixture distributions. *Technometrics*, **37**, 185–194.

Johns, M.V. (1988) Importance sampling for bootstrap confidence intervals. *Journal of the American Statistical Association*, **83**, 709–714.

See Also

boot, exp.tilt, imp.weights, smooth.f, tilt.boot

Examples

```
# Example 9.8 of Davison and Hinkley (1997) requires tilting the
# resampling distribution of the studentized statistic to be centred
# at the observed value of the test statistic, 1.84.  In this example
# we show how certain estimates can be found using resamples taken from
# the tilted distribution.
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2] - mns[1], s2hat, (mns[2] - mns[1] - orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.L <- empinf(data = grav1, statistic = grav.fun, stype = "w",
                 strata = grav1[,2], index = 3, orig = grav.z0[1])
grav.tilt <- exp.tilt(grav.L, grav.z0[3], strata = grav1[, 2])
grav.tilt.boot <- boot(grav1, grav.fun, R = 199, stype = "w",
                       strata = grav1[, 2], weights = grav.tilt$p,
                       orig = grav.z0[1])
# Since the weights are needed for all calculations, we shall calculate
# them once only.
grav.w <- imp.weights(grav.tilt.boot)
grav.mom <- imp.moments(grav.tilt.boot, w = grav.w, index = 3)
grav.p <- imp.prob(grav.tilt.boot, w = grav.w, index = 3, t0 = grav.z0[3])
unlist(grav.p)
grav.q <- imp.quantile(grav.tilt.boot, w = grav.w, index = 3,
                       alpha = c(0.9, 0.95, 0.975, 0.99))
as.data.frame(grav.q)
```

---

imp.weights *Importance Sampling Weights*

Description

This function calculates the importance sampling weight required to correct for simulation from a distribution with probabilities p when estimates are required assuming that simulation was from an alternative distribution with probabilities q.

Usage

```
imp.weights(boot.out, def = TRUE, q = NULL)
```

Arguments

boot.out	A object of class "boot" generated by boot or tilt.boot. Typically the bootstrap simulations would have been done using importance resampling and we wish to do our calculations under the assumption of sampling with equal probabilities.
def	A logical variable indicating whether the defensive mixture distribution weights should be calculated. This makes sense only in the case where the replicates in boot.out were simulated under a number of different distributions. If this is the case then the defensive mixture weights use a mixture of the distributions used in the bootstrap. The alternative is to calculate the weights for each replicate using knowledge of the distribution from which the bootstrap resample was generated.
q	A vector of probabilities specifying the resampling distribution from which we require inferences to be made. In general this would correspond to the usual bootstrap resampling distribution which gives equal weight to each of the original observations and this is the default. q must have length equal to the number of observations in the boot.out$data and all elements of q must be positive.

Details

The importance sampling weight for a bootstrap replicate with frequency vector f is given by prod((q/p)^f). This reweights the replicates so that estimates can be found as if the bootstrap resamples were generated according to the probabilities q even though, in fact, they came from the distribution p.

Value

A vector of importance weights of the same length as boot.out$t. These weights can then be used to reweight boot.out$t so that estimates can be found as if the simulations were from a distribution with probabilities q.

Note

See the example in the help for imp.moments for an example of using imp.weights.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Hesterberg, T. (1995) Weighted average importance sampling and defensive mixture distributions. *Technometrics*, **37**, 185–194.

Johns, M.V. (1988) Importance sampling for bootstrap confidence intervals. *Journal of the American Statistical Association*, **83**, 709–714.

See Also

boot, exp.tilt, imp.moments, smooth.f, tilt.boot

---

inv.logit *Inverse Logit Function*

Description

Given a numeric object return the inverse logit of the values.

Usage

```
inv.logit(x)
```

Arguments

x	A numeric object. Missing values (NAs) are allowed.

Details

The inverse logit is defined by exp(x)/(1+exp(x)). Values in x of -Inf or Inf return logits of 0 or 1 respectively. Any NAs in the input will also be NAs in the output.

Value

An object of the same type as x containing the inverse logits of the input values.

See Also

logit, plogis for which this is a wrapper.

---

islay *Jura Quartzite Azimuths on Islay*

Description

The islay data frame has 18 rows and 1 columns.

Measurements were taken of paleocurrent azimuths from the Jura Quartzite on the Scottish island of Islay.

Usage

```
islay
```

Format

This data frame contains the following column:

`theta` The angle of the azimuth in degrees East of North.

Source

The data were obtained from

Hand, D.J., Daly, F., Lunn, A.D., McConway, K.J. and Ostrowski, E. (1994) *A Handbook of Small Data Sets*, Chapman and Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Till, R. (1974) *Statistical Methods for the Earth Scientist*. Macmillan.

---

jack.after.boot *Jackknife-after-Bootstrap Plots*

Description

This function calculates the jackknife influence values from a bootstrap output object and plots the corresponding jackknife-after-bootstrap plot.

Usage

```
jack.after.boot(boot.out, index = 1, t = NULL, L = NULL,
                useJ = TRUE, stinf = TRUE, alpha = NULL,
                main = "", ylab = NULL, ...)
```

Arguments

`boot.out`	An object of class `"boot"` which would normally be created by a call to `boot`. It should represent a nonparametric bootstrap. For reliable results `boot.out$R` should be reasonably large.
`index`	The index of the statistic of interest in the output of `boot.out$statistic`.
`t`	A vector of length `boot.out$R` giving the bootstrap replicates of the statistic of interest. This is useful if the statistic of interest is a function of the calculated bootstrap output. If it is not supplied then the default is `boot.out$t[,index]`.
`L`	The empirical influence values for the statistic of interest. These are used only if `useJ` is `FALSE`. If they are not supplied and are needed, they are calculated by a call to `empinf`. If `L` is supplied then it is assumed that they are the infinitesimal jackknife values.

useJ	A logical variable indicating if the jackknife influence values calculated from the bootstrap replicates should be used. If FALSE the empirical influence values are used. The default is TRUE.
stinf	A logical variable indicating whether to standardize the jackknife values before plotting them. If TRUE then the jackknife values used are divided by their standard error.
alpha	The quantiles at which the plots are required. The default is c(0.05, 0.1, 0.16, 0.5, 0.84, 0.9, 0.9
main	A character string giving the main title for the plot.
ylab	The label for the Y axis. If the default values of alpha are used and ylab is not supplied then a label indicating which percentiles are plotted is used. If alpha is supplied then the default label will not say which percentiles were used.
...	Any extra arguments required by boot.out$statistic. These are required only if useJ is FALSE and L is not supplied, in which case they are passed to empinf for use in calculation of the empirical influence values.

Details

The centred jackknife quantiles for each observation are estimated from those bootstrap samples in which the particular observation did not appear. These are then plotted against the influence values. If useJ is TRUE then the influence values are found in the same way as the difference between the mean of the statistic in the samples excluding the observations and the mean in all samples. If useJ is FALSE then empirical influence values are calculated by calling empinf.

The resulting plots are useful diagnostic tools for looking at the way individual observations affect the bootstrap output.

The plot will consist of a number of horizontal dotted lines which correspond to the quantiles of the centred bootstrap distribution. For each data point the quantiles of the bootstrap distribution calculated by omitting that point are plotted against the (possibly standardized) jackknife values. The observation number is printed below the plots. To make it easier to see the effect of omitting points on quantiles, the plotted quantiles are joined by line segments. These plots provide a useful diagnostic tool in establishing the effect of individual observations on the bootstrap distribution. See the references below for some guidelines on the interpretation of the plots.

Value

There is no returned value but a plot is generated on the current graphics display.

Side Effects

A plot is created on the current graphics device.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, **54**, 83–127.

See Also

boot, empinf

Examples

```
#  To draw the jackknife-after-bootstrap plot for the head size data as in
#  Example 3.24 of Davison and Hinkley (1997)
frets.fun <- function(data, i) {
    pcorr <- function(x) {
    #  Function to find the correlations and partial correlations between
    #  the four measurements.
         v <- cor(x)
         v.d <- diag(var(x))
         iv <- solve(v)
         iv.d <- sqrt(diag(iv))
         iv <- - diag(1/iv.d) %*% iv %*% diag(1/iv.d)
         q <- NULL
         n <- nrow(v)
         for (i in 1:(n-1))
              q <- rbind( q, c(v[i, 1:i], iv[i,(i+1):n]) )
         q <- rbind( q, v[n, ] )
         diag(q) <- round(diag(q))
         q
    }
    d <- data[i, ]
    v <- pcorr(d)
    c(v[1,], v[2,], v[3,], v[4,])
}
frets.boot <- boot(log(as.matrix(frets)), frets.fun, R = 999)
#  we will concentrate on the partial correlation between head breadth
#  for the first son and head length for the second.  This is the 7th
#  element in the output of frets.fun so we set index = 7
jack.after.boot(frets.boot, useJ = FALSE, stinf = FALSE, index = 7)
```

---

k3.linear *Linear Skewness Estimate*

Description

Estimates the skewness of a statistic from its empirical influence values.

Usage

```
k3.linear(L, strata = NULL)
```

Arguments

L	Vector of the empirical influence values of a statistic. These will usually be calculated by a call to empinf.

strata — A numeric vector or factor specifying which observations (and hence which components of L) come from which strata.

Value

The skewness estimate calculated from L.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

empinf, linear.approx, var.linear

Examples

```
#  To estimate the skewness of the ratio of means for the city data.
ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
k3.linear(empinf(data = city, statistic = ratio))
```

---

linear.approx — *Linear Approximation of Bootstrap Replicates*

Description

This function takes a bootstrap object and for each bootstrap replicate it calculates the linear approximation to the statistic of interest for that bootstrap sample.

Usage

```
linear.approx(boot.out, L = NULL, index = 1, type = NULL,
              t0 = NULL, t = NULL, ...)
```

Arguments

boot.out — An object of class "boot" representing a nonparametric bootstrap. It will usually be created by the function boot.

L — A vector containing the empirical influence values for the statistic of interest. If it is not supplied then L is calculated through a call to empinf.

index — The index of the variable of interest within the output of boot.out$statistic.

type — This gives the type of empirical influence values to be calculated. It is not used if L is supplied. The possible types of empirical influence values are described in the help for empinf.

`t0`	The observed value of the statistic of interest. The input value is used only if one of `t` or `L` is also supplied. The default value is `boot.out$t0[index]`. If `t0` is supplied but neither `t` nor `L` are supplied then `t0` is set to `boot.out$t0[index]` and a warning is generated.
`t`	A vector of bootstrap replicates of the statistic of interest. If `t0` is missing then `t` is not used, otherwise it is used to calculate the empirical influence values (if they are not supplied in `L`).
`...`	Any extra arguments required by `boot.out$statistic`. These are needed if `L` is not supplied as they are used by `empinf` to calculate empirical influence values.

Details

The linear approximation to a bootstrap replicate with frequency vector `f` is given by `t0 + sum(L * f)/n` in the one sample with an easy extension to the stratified case. The frequencies are found by calling `boot.array`.

Value

A vector of length `boot.out$R` with the linear approximations to the statistic of interest for each of the bootstrap samples.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

`boot`, `empinf`, `control`

Examples

```
# Using the city data let us look at the linear approximation to the
# ratio statistic and its logarithm. We compare these with the
# corresponding plots for the bigcity data

ratio <- function(d, w) sum(d$x * w)/sum(d$u * w)
city.boot <- boot(city, ratio, R = 499, stype = "w")
bigcity.boot <- boot(bigcity, ratio, R = 499, stype = "w")
op <- par(pty = "s", mfrow = c(2, 2))

# The first plot is for the city data ratio statistic.
city.lin1 <- linear.approx(city.boot)
lim <- range(c(city.boot$t,city.lin1))
plot(city.boot$t, city.lin1, xlim = lim, ylim = lim,
     main = "Ratio; n=10", xlab = "t*", ylab = "tL*")
abline(0, 1)

# Now for the log of the ratio statistic for the city data.
```

```
city.lin2 <- linear.approx(city.boot,t0 = log(city.boot$t0),
                           t = log(city.boot$t))
lim <- range(c(log(city.boot$t),city.lin2))
plot(log(city.boot$t), city.lin2, xlim = lim, ylim = lim,
     main = "Log(Ratio); n=10", xlab = "t*", ylab = "tL*")
abline(0, 1)

# The ratio statistic for the bigcity data.
bigcity.lin1 <- linear.approx(bigcity.boot)
lim <- range(c(bigcity.boot$t,bigcity.lin1))
plot(bigcity.lin1, bigcity.boot$t, xlim = lim, ylim = lim,
     main = "Ratio; n=49", xlab = "t*", ylab = "tL*")
abline(0, 1)

# Finally the log of the ratio statistic for the bigcity data.
bigcity.lin2 <- linear.approx(bigcity.boot,t0 = log(bigcity.boot$t0),
                              t = log(bigcity.boot$t))
lim <- range(c(log(bigcity.boot$t),bigcity.lin2))
plot(bigcity.lin2, log(bigcity.boot$t), xlim = lim, ylim = lim,
     main = "Log(Ratio); n=49", xlab = "t*", ylab = "tL*")
abline(0, 1)

par(op)
```

`lines.saddle.distn` *Add a Saddlepoint Approximation to a Plot*

Description

This function adds a line corresponding to a saddlepoint density or distribution function approximation to the current plot.

Usage

```
## S3 method for class 'saddle.distn'
lines(x, dens = TRUE, h = function(u) u, J = function(u) 1,
      npts = 50, lty = 1, ...)
```

Arguments

`x`	An object of class `"saddle.distn"` (see `saddle.distn.object` representing a saddlepoint approximation to a distribution.
`dens`	A logical variable indicating whether the saddlepoint density (`TRUE`; the default) or the saddlepoint distribution function (`FALSE`) should be plotted.
`h`	Any transformation of the variable that is required. Its first argument must be the value at which the approximation is being performed and the function must be vectorized.

J	When dens=TRUE this function specifies the Jacobian for any transformation that may be necessary. The first argument of J must the value at which the approximation is being performed and the function must be vectorized. If h is supplied J must also be supplied and both must have the same argument list.
npts	The number of points to be used for the plot. These points will be evenly spaced over the range of points used in finding the saddlepoint approximation.
lty	The line type to be used.
...	Any additional arguments to h and J.

Details

The function uses smooth.spline to produce the saddlepoint curve. When dens=TRUE the spline is on the log scale and when dens=FALSE it is on the probit scale.

Value

sad.d is returned invisibly.

Side Effects

A line is added to the current plot.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

saddle.distn

Examples

```
# In this example we show how a plot such as that in Figure 9.9 of
# Davison and Hinkley (1997) may be produced.  Note the large number of
# bootstrap replicates required in this example.
expdata <- rexp(12)
vfun <- function(d, i) {
     n <- length(d)
     (n-1)/n*var(d[i])
}
exp.boot <- boot(expdata,vfun, R = 9999)
exp.L <- (expdata - mean(expdata))^2 - exp.boot$t0
exp.tL <- linear.approx(exp.boot, L = exp.L)
hist(exp.tL, nclass = 50, probability = TRUE)
exp.t0 <- c(0, sqrt(var(exp.boot$t)))
exp.sp <- saddle.distn(A = exp.L/12,wdist = "m", t0 = exp.t0)

# The saddlepoint approximation in this case is to the density of
# t-t0 and so t0 must be added for the plot.
```

```
lines(exp.sp, h = function(u, t0) u+t0, J = function(u, t0) 1,
      t0 = exp.boot$t0)
```

---

logit *Logit of Proportions*

Description

This function calculates the logit of proportions.

Usage

```
logit(p)
```

Arguments

p	A numeric Splus object, all of whose values are in the range [0,1]. Missing values (NAs) are allowed.

Details

If any elements of p are outside the unit interval then an error message is generated. Values of p equal to 0 or 1 (to within machine precision) will return -Inf or Inf respectively. Any NAs in the input will also be NAs in the output.

Value

A numeric object of the same type as p containing the logits of the input values.

See Also

inv.logit, qlogis for which this is a wrapper.

---

manaus *Average Heights of the Rio Negro river at Manaus*

Description

The manaus time series is of class "ts" and has 1080 observations on one variable.

The data values are monthly averages of the daily stages (heights) of the Rio Negro at Manaus. Manaus is 18km upstream from the confluence of the Rio Negro with the Amazon but because of the tiny slope of the water surface and the lower courses of its flatland affluents, they may be regarded as a good approximation of the water level in the Amazon at the confluence. The data here cover 90 years from January 1903 until December 1992.

The Manaus gauge is tied in with an arbitrary bench mark of 100m set in the steps of the Municipal Prefecture; gauge readings are usually referred to sea level, on the basis of a mark on the steps

leading to the Parish Church (Matriz), which is assumed to lie at an altitude of 35.874 m according to observations made many years ago under the direction of Samuel Pereira, an engineer in charge of the Manaus Sanitation Committee Whereas such an altitude cannot, by any means, be considered to be a precise datum point, observations have been provisionally referred to it. The measurements are in metres.

Source

The data were kindly made available by Professors H. O'Reilly Sternberg and D. R. Brillinger of the University of California at Berkeley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Sternberg, H. O'R. (1987) Aggravation of floods in the Amazon river as a consequence of deforestation? *Geografiska Annaler*, **69A**, 201-219.

Sternberg, H. O'R. (1995) Waters and wetlands of Brazilian Amazonia: An uncertain future. In *The Fragile Tropics of Latin America: Sustainable Management of Changing Environments*, Nishizawa, T. and Uitto, J.I. (editors), United Nations University Press, 113-179.

`melanoma` *Survival from Malignant Melanoma*

Description

The `melanoma` data frame has 205 rows and 7 columns.

The data consist of measurements made on patients with malignant melanoma. Each patient had their tumour removed by surgery at the Department of Plastic Surgery, University Hospital of Odense, Denmark during the period 1962 to 1977. The surgery consisted of complete removal of the tumour together with about 2.5cm of the surrounding skin. Among the measurements taken were the thickness of the tumour and whether it was ulcerated or not. These are thought to be important prognostic variables in that patients with a thick and/or ulcerated tumour have an increased chance of death from melanoma. Patients were followed until the end of 1977.

Usage

```
melanoma
```

Format

This data frame contains the following columns:

`time` Survival time in days since the operation, possibly censored.

`status` The patients status at the end of the study. 1 indicates that they had died from melanoma, 2 indicates that they were still alive and 3 indicates that they had died from causes unrelated to their melanoma.

`sex` The patients sex; 1=male, 0=female.

`age` Age in years at the time of the operation.

`year` Year of operation.

`thickness` Tumour thickness in mm.

`ulcer` Indicator of ulceration; 1=present, 0=absent.

Note

This dataset is not related to the dataset in the **lattice** package with the same name.

Source

The data were obtained from

Andersen, P.K., Borgan, O., Gill, R.D. and Keiding, N. (1993) *Statistical Models Based on Counting Processes*. Springer-Verlag.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Venables, W.N. and Ripley, B.D. (1994) *Modern Applied Statistics with S-Plus*. Springer-Verlag.

`motor` *Data from a Simulated Motorcycle Accident*

Description

The `motor` data frame has 94 rows and 4 columns. The rows are obtained by removing replicate values of `time` from the dataset `mcycle`. Two extra columns are added to allow for strata with a different residual variance in each stratum.

Usage

```
motor
```

Format

This data frame contains the following columns:

`times` The time in milliseconds since impact.

`accel` The recorded head acceleration (in g).

`strata` A numeric column indicating to which of the three strata (numbered 1, 2 and 3) the observations belong.

`v` An estimate of the residual variance for the observation. `v` is constant within the strata but a different estimate is used for each of the three strata.

Source

The data were obtained from

Silverman, B.W. (1985) Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society, B*, **47**, 1–52.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Venables, W.N. and Ripley, B.D. (1994) *Modern Applied Statistics with S-Plus*. Springer-Verlag.

See Also

`mcycle`

---

neuro *Neurophysiological Point Process Data*

Description

`neuro` is a matrix containing times of observed firing of a neuron in windows of 250ms either side of the application of a stimulus to a human subject. Each row of the matrix is a replication of the experiment and there were a total of 469 replicates.

Note

There are a lot of missing values in the matrix as different numbers of firings were observed in different replicates. The number of firings observed varied from 2 to 6.

Source

The data were collected and kindly made available by Dr. S.J. Boniface of the Neurophysiology Unit at the Radcliffe Infirmary, Oxford.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Ventura, V., Davison, A.C. and Boniface, S.J. (1997) A stochastic model for the effect of magnetic brain stimulation on a motorneurone. To appear in *Applied Statistics*.

`nitrofen` *Toxicity of Nitrofen in Aquatic Systems*

Description

The `nitrofen` data frame has 50 rows and 5 columns.

Nitrofen is a herbicide that was used extensively for the control of broad-leaved and grass weeds in cereals and rice. Although it is relatively non-toxic to adult mammals, nitrofen is a significant tetragen and mutagen. It is also acutely toxic and reproductively toxic to cladoceran zooplankton. Nitrofen is no longer in commercial use in the U.S., having been the first pesticide to be withdrawn due to tetragenic effects.

The data here come from an experiment to measure the reproductive toxicity of nitrofen on a species of zooplankton (*Ceriodaphnia dubia*). 50 animals were randomized into batches of 10 and each batch was put in a solution with a measured concentration of nitrofen. Then the number of live offspring in each of the three broods to each animal was recorded.

Usage

```
nitrofen
```

Format

This data frame contains the following columns:

`conc` The nitrofen concentration in the solution (mug/litre).

`brood1` The number of live offspring in the first brood.

`brood2` The number of live offspring in the second brood.

`brood3` The number of live offspring in the third brood.

`total` The total number of live offspring in the first three broods.

Source

The data were obtained from

Bailer, A.J. and Oris, J.T. (1994) Assessing toxicity of pollutants in aquatic systems. In *Case Studies in Biometry*. N. Lange, L. Ryan, L. Billard, D. Brillinger, L. Conquest and J. Greenhouse (editors), 25–40. John Wiley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`nodal` *Nodal Involvement in Prostate Cancer*

Description

The `nodal` data frame has 53 rows and 7 columns.

The treatment strategy for a patient diagnosed with cancer of the prostate depend highly on whether the cancer has spread to the surrounding lymph nodes. It is common to operate on the patient to get samples from the nodes which can then be analysed under a microscope but clearly it would be preferable if an accurate assessment of nodal involvement could be made without surgery.

For a sample of 53 prostate cancer patients, a number of possible predictor variables were measured before surgery. The patients then had surgery to determine nodal involvement. It was required to see if nodal involvement could be accurately predicted from the predictor variables and which ones were most important.

Usage

```
nodal
```

Format

This data frame contains the following columns:

`m` A column of ones.

`r` An indicator of nodal involvement.

`aged` The patients age dichotomized into less than 60 (`0`) and 60 or over `1`.

`stage` A measurement of the size and position of the tumour observed by palpitation with the fingers via the rectum. A value of `1` indicates a more serious case of the cancer.

`grade` Another indicator of the seriousness of the cancer, this one is determined by a pathology reading of a biopsy taken by needle before surgery. A value of `1` indicates a more serious case of the cancer.

`xray` A third measure of the seriousness of the cancer taken from an X-ray reading. A value of `1` indicates a more serious case of the cancer.

`acid` The level of acid phosphatase in the blood serum.

Source

The data were obtained from

Brown, B.W. (1980) Prediction analysis for binary data. In *Biostatistics Casebook*. R.G. Miller, B. Efron, B.W. Brown and L.E. Moses (editors), 3–18. John Wiley.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

norm.ci *Normal Approximation Confidence Intervals*

Description

Using the normal approximation to a statistic, calculate equi-tailed two-sided confidence intervals.

Usage

```
norm.ci(boot.out = NULL, conf = 0.95, index = 1, var.t0 = NULL,
        t0 = NULL, t = NULL, L = NULL, h = function(t) t,
        hdot = function(t) 1, hinv = function(t) t)
```

Arguments

boot.out	A bootstrap output object returned from a call to boot. If t0 is missing then boot.out is a required argument. It is also required if both var.t0 and t are missing.
conf	A scalar or vector containing the confidence level(s) of the required interval(s).
index	The index of the statistic of interest within the output of a call to boot.out$statistic. It is not used if boot.out is missing, in which case t0 must be supplied.
var.t0	The variance of the statistic of interest. If it is not supplied then var(t) is used.
t0	The observed value of the statistic of interest. If it is missing then it is taken from boot.out which is required in that case.
t	Bootstrap replicates of the variable of interest. These are used to estimate the variance of the statistic of interest if var.t0 is not supplied. The default value is boot.out$t[,index].
L	The empirical influence values for the statistic of interest. These are used to calculate var.t0 if neither var.t0 nor boot.out are supplied. If a transformation is supplied through h then the influence values must be for the untransformed statistic t0.
h	A function defining a monotonic transformation, the intervals are calculated on the scale of h(t) and the inverse function hinv is applied to the resulting intervals. h must be a function of one variable only and must be vectorized. The default is the identity function.
hdot	A function of one argument returning the derivative of h. It is a required argument if h is supplied and is used for approximating the variance of h(t0). The default is the constant function 1.
hinv	A function, like h, which returns the inverse of h. It is used to transform the intervals calculated on the scale of h(t) back to the original scale. The default is the identity function. If h is supplied but hinv is not, then the intervals returned will be on the transformed scale.

Details

It is assumed that the statistic of interest has an approximately normal distribution with variance `var.t0` and so a confidence interval of length `2*qnorm((1+conf)/2)*sqrt(var.t0)` is found. If `boot.out` or `t` are supplied then the interval is bias-corrected using the bootstrap bias estimate, and so the interval would be centred at `2*t0-mean(t)`. Otherwise the interval is centred at `t0`.

Value

If `length(conf)` is 1 then a vector containing the confidence level and the endpoints of the interval is returned. Otherwise, the returned value is a matrix where each row corresponds to a different confidence level.

Note

This function is primarily designed to be called by `boot.ci` to calculate the normal approximation after a bootstrap but it can also be used without doing any bootstrap calculations as long as `t0` and `var.t0` can be supplied. See the examples below.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

`boot.ci`

Examples

```
#  In Example 5.1 of Davison and Hinkley (1997), normal approximation
#  confidence intervals are found for the air-conditioning data.
air.mean <- mean(aircondit$hours)
air.n <- nrow(aircondit)
air.v <- air.mean^2/air.n
norm.ci(t0 = air.mean, var.t0 = air.v)
exp(norm.ci(t0 = log(air.mean), var.t0 = 1/air.n)[2:3])

# Now a more complicated example - the ratio estimate for the city data.
ratio <- function(d, w)
     sum(d$x * w)/sum(d$u *w)
city.v <- var.linear(empinf(data = city, statistic = ratio))
norm.ci(t0 = ratio(city,rep(0.1,10)), var.t0 = city.v)
```

`nuclear` *Nuclear Power Station Construction Data*

Description

The `nuclear` data frame has 32 rows and 11 columns.

The data relate to the construction of 32 light water reactor (LWR) plants constructed in the U.S.A in the late 1960's and early 1970's. The data was collected with the aim of predicting the cost of construction of further LWR plants. 6 of the power plants had partial turnkey guarantees and it is possible that, for these plants, some manufacturers' subsidies may be hidden in the quoted capital costs.

Usage

```
nuclear
```

Format

This data frame contains the following columns:

`cost` The capital cost of construction in millions of dollars adjusted to 1976 base.

`date` The date on which the construction permit was issued. The data are measured in years since January 1 1990 to the nearest month.

`t1` The time between application for and issue of the construction permit.

`t2` The time between issue of operating license and construction permit.

`cap` The net capacity of the power plant (MWe).

`pr` A binary variable where `1` indicates the prior existence of a LWR plant at the same site.

`ne` A binary variable where `1` indicates that the plant was constructed in the north-east region of the U.S.A.

`ct` A binary variable where `1` indicates the use of a cooling tower in the plant.

`bw` A binary variable where `1` indicates that the nuclear steam supply system was manufactured by Babcock-Wilcox.

`cum.n` The cumulative number of power plants constructed by each architect-engineer.

`pt` A binary variable where `1` indicates those plants with partial turnkey guarantees.

Source

The data were obtained from

Cox, D.R. and Snell, E.J. (1981) *Applied Statistics: Principles and Examples*. Chapman and Hall.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

paulsen *Neurotransmission in Guinea Pig Brains*

Description

The paulsen data frame has 346 rows and 1 columns.

Sections were prepared from the brain of adult guinea pigs. Spontaneous currents that flowed into individual brain cells were then recorded and the peak amplitude of each current measured. The aim of the experiment was to see if the current flow was quantal in nature (i.e. that it is not a single burst but instead is built up of many smaller bursts of current). If the current was indeed quantal then it would be expected that the distribution of the current amplitude would be multimodal with modes at regular intervals. The modes would be expected to decrease in magnitude for higher current amplitudes.

Usage

```
paulsen
```

Format

This data frame contains the following column:

y The current flowing into individual brain cells. The currents are measured in pico-amperes.

Source

The data were kindly made available by Dr. O. Paulsen from the Department of Pharmacology at the University of Oxford.

Paulsen, O. and Heggelund, P. (1994) The quantal size at retinogeniculate synapses determined from spontaneous and evoked EPSCs in guinea-pig thalamic slices. *Journal of Physiology*, **480**, 505–511.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

plot.boot *Plots of the Output of a Bootstrap Simulation*

Description

This takes a bootstrap object and produces plots for the bootstrap replicates of the variable of interest.

Usage

```
## S3 method for class 'boot'
plot(x, index = 1, t0 = NULL, t = NULL, jack = FALSE,
     qdist = "norm", nclass = NULL, df, ...)
```

Arguments

x	An object of class "boot" returned from one of the bootstrap generation functions.
index	The index of the variable of interest within the output of boot.out. This is ignored if t and t0 are supplied.
t0	The original value of the statistic. This defaults to boot.out$t0[index] unless t is supplied when it defaults to NULL. In that case no vertical line is drawn on the histogram.
t	The bootstrap replicates of the statistic. Usually this will take on its default value of boot.out$t[,index], however it may be useful sometimes to supply a different set of values which are a function of boot.out$t.
jack	A logical value indicating whether a jackknife-after-bootstrap plot is required. The default is not to produce such a plot.
qdist	The distribution against which the Q-Q plot should be drawn. At present "norm" (normal distribution - the default) and "chisq" (chi-squared distribution) are the only possible values.
nclass	An integer giving the number of classes to be used in the bootstrap histogram. The default is the integer between 10 and 100 closest to ceiling(length(t)/25).
df	If qdist is "chisq" then this is the degrees of freedom for the chi-squared distribution to be used. It is a required argument in that case.
...	When jack is TRUE additional parameters to jack.after.boot can be supplied. See the help file for jack.after.boot for details of the possible parameters.

Details

This function will generally produce two side-by-side plots. The left plot will be a histogram of the bootstrap replicates. Usually the breaks of the histogram will be chosen so that t0 is at a breakpoint and all intervals are of equal length. A vertical dotted line indicates the position of t0. This cannot be done if t is supplied but t0 is not and so, in that case, the breakpoints are computed by hist using the nclass argument and no vertical line is drawn.

The second plot is a Q-Q plot of the bootstrap replicates. The order statistics of the replicates can be plotted against normal or chi-squared quantiles. In either case the expected line is also plotted. For the normal, this will have intercept `mean(t)` and slope `sqrt(var(t))` while for the chi-squared it has intercept 0 and slope 1.

If `jack` is `TRUE` a third plot is produced beneath these two. That plot is the jackknife-after-bootstrap plot. This plot may only be requested when nonparametric simulation has been used. See `jack.after.boot` for further details of this plot.

Value

`boot.out` is returned invisibly.

Side Effects

All screens are closed and cleared and a number of plots are produced on the current graphics device. Screens are closed but not cleared at termination of this function.

See Also

`boot`, `jack.after.boot`, `print.boot`

Examples

```
# We fit an exponential model to the air-conditioning data and use
# that for a parametric bootstrap.  Then we look at plots of the
# resampled means.
air.rg <- function(data, mle) rexp(length(data), 1/mle)

air.boot <- boot(aircondit$hours, mean, R = 999, sim = "parametric",
                 ran.gen = air.rg, mle = mean(aircondit$hours))
plot(air.boot)

# In the difference of means example for the last two series of the
# gravity data
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
grav.fun <- function(dat, w) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2] - mns[1], s2hat)
}

grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w", strata = grav1[, 2])
plot(grav.boot)
# now suppose we want to look at the studentized differences.
grav.z <- (grav.boot$t[, 1]-grav.boot$t0[1])/sqrt(grav.boot$t[, 2])
plot(grav.boot, t = grav.z, t0 = 0)
```

```
# In this example we look at the one of the partial correlations for the
# head dimensions in the dataset frets.
frets.fun <- function(data, i) {
    pcorr <- function(x) {
    #  Function to find the correlations and partial correlations between
    #  the four measurements.
         v <- cor(x)
         v.d <- diag(var(x))
         iv <- solve(v)
         iv.d <- sqrt(diag(iv))
         iv <- - diag(1/iv.d) %*% iv %*% diag(1/iv.d)
         q <- NULL
         n <- nrow(v)
         for (i in 1:(n-1))
              q <- rbind( q, c(v[i, 1:i], iv[i,(i+1):n]) )
         q <- rbind( q, v[n, ] )
         diag(q) <- round(diag(q))
         q
    }
    d <- data[i, ]
    v <- pcorr(d)
    c(v[1,], v[2,], v[3,], v[4,])
}
frets.boot <- boot(log(as.matrix(frets)),  frets.fun,  R = 999)
plot(frets.boot, index = 7, jack = TRUE, stinf = FALSE, useJ = FALSE)
```

`poisons` *Animal Survival Times*

Description

The `poisons` data frame has 48 rows and 3 columns.

The data form a 3x4 factorial experiment, the factors being three poisons and four treatments. Each combination of the two factors was used for four animals, the allocation to animals having been completely randomized.

Usage

```
poisons
```

Format

This data frame contains the following columns:

`time` The survival time of the animal in units of 10 hours.

`poison` A factor with levels `1`, `2` and `3` giving the type of poison used.

`treat` A factor with levels `A`, `B`, `C` and `D` giving the treatment.

Source

The data were obtained from

Box, G.E.P. and Cox, D.R. (1964) An analysis of transformations (with Discussion). *Journal of the Royal Statistical Society, B*, **26**, 211–252.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`polar` *Pole Positions of New Caledonian Laterites*

Description

The `polar` data frame has 50 rows and 2 columns.

The data are the pole positions from a paleomagnetic study of New Caledonian laterites.

Usage

```
polar
```

Format

This data frame contains the following columns:

`lat` The latitude (in degrees) of the pole position. Note that all latitudes are negative as the axis is taken to be in the lower hemisphere.

`long` The longitude (in degrees) of the pole position.

Source

The data were obtained from

Fisher, N.I., Lewis, T. and Embleton, B.J.J. (1987) *Statistical Analysis of Spherical Data*. Cambridge University Press.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

print.boot *Print a Summary of a Bootstrap Object*

Description

This is a method for the function `print()` for objects of the class `"boot"`.

Usage

```
## S3 method for class 'boot'
print(x, digits = getOption("digits"),
      index = 1:ncol(boot.out$t), ...)
```

Arguments

`x`	A bootstrap output object of class `"boot"` generated by one of the bootstrap functions.
`digits`	The number of digits to be printed in the summary statistics.
`index`	Indices indicating for which elements of the bootstrap output summary statistics are required.
`...`	further arguments passed to or from other methods.

Details

For each statistic calculated in the bootstrap the original value and the bootstrap estimates of its bias and standard error are printed. If `boot.out$t0` is missing (such as when it was created by a call to `tsboot` with `orig.t=FALSE`) the bootstrap mean and standard error are printed. If resampling was done using importance resampling weights, then the bootstrap estimates are reweighted as if uniform resampling had been done. The ratio importance sampling estimates are used and if there were a number of distributions then defensive mixture distributions are used. In this case an extra column with the mean of the observed bootstrap statistics is also printed.

Value

The bootstrap object is returned invisibly.

See Also

`boot`, `censboot`, `imp.moments`, `plot.boot`, `tilt.boot`, `tsboot`

print.bootci *Print Bootstrap Confidence Intervals*

Description

This is a method for the function print() to print objects of the class "bootci".

Usage

```
## S3 method for class 'bootci'
print(x, hinv = NULL, ...)
```

Arguments

x	The output from a call to boot.ci.
hinv	A transformation to be made to the interval end-points before they are printed.
...	further arguments passed to or from other methods.

Details

This function prints out the results from boot.ci in a "nice" format. It also notes whether the scale of the intervals is the original scale of the input to boot.ci or a different scale and whether the calculations were done on a transformed scale. It also looks at the order statistics that were used in calculating the intervals. If the smallest or largest values were used then it prints a message

```
Warning : Intervals used Extreme Quantiles
```

Such intervals should be considered very unstable and not relied upon for inferences. Even if the extreme values are not used, it is possible that the intervals are unstable if they used quantiles close to the extreme values. The function alerts the user to intervals which use the upper or lower 10 order statistics with the message

```
Some intervals may be unstable
```

Value

The object ci.out is returned invisibly.

See Also

boot.ci

print.saddle.distn *Print Quantiles of Saddlepoint Approximations*

Description

This is a method for the function `print()` to print objects of class `"saddle.distn"`.

Usage

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

Arguments

`x`	An object of class `"saddle.distn"` created by a call to `saddle.distn`.
`...`	further arguments passed to or from other methods.

Details

The quantiles of the saddlepoint approximation to the distribution are printed along with the original call and some other useful information.

Value

The input is returned invisibly.

See Also

`lines.saddle.distn`, `saddle.distn`

print.simplex *Print Solution to Linear Programming Problem*

Description

This is a method for the function `print()` to print objects of class `"simplex"`.

Usage

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

Arguments

`x`	An object of class `"simplex"` created by calling the function `simplex` to solve a linear programming problem.
`...`	further arguments passed to or from other methods.

Details

The coefficients of the objective function are printed. If a solution to the linear programming problem was found then the solution and the optimal value of the objective function are printed. If a feasible solution was found but the maximum number of iterations was exceeded then the last feasible solution and the objective function value at that point are printed. If no feasible solution could be found then a message stating that is printed.

Value

`x` is returned silently.

See Also

`simplex`

`remission` *Cancer Remission and Cell Activity*

Description

The `remission` data frame has 27 rows and 3 columns.

Usage

```
remission
```

Format

This data frame contains the following columns:

`LI` A measure of cell activity.

`m` The number of patients in each group (all values are actually 1 here).

`r` The number of patients (out of `m`) who went into remission.

Source

The data were obtained from

Freeman, D.H. (1987) *Applied Categorical Data Analysis*. Marcel Dekker.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

saddle *Saddlepoint Approximations for Bootstrap Statistics*

Description

This function calculates a saddlepoint approximation to the distribution of a linear combination of **W** at a particular point u, where **W** is a vector of random variables. The distribution of **W** may be multinomial (default), Poisson or binary. Other distributions are possible also if the adjusted cumulant generating function and its second derivative are given. Conditional saddlepoint approximations to the distribution of one linear combination given the values of other linear combinations of **W** can be calculated for **W** having binary or Poisson distributions.

Usage

```
saddle(A = NULL, u = NULL, wdist = "m", type = "simp", d = NULL,
       d1 = 1, init = rep(0.1, d), mu = rep(0.5, n), LR = FALSE,
       strata = NULL, K.adj = NULL, K2 = NULL)
```

Arguments

A	A vector or matrix of known coefficients of the linear combinations of **W**. It is a required argument unless K.adj and K2 are supplied, in which case it is ignored.
u	The value at which it is desired to calculate the saddlepoint approximation to the distribution of the linear combination of **W**. It is a required argument unless K.adj and K2 are supplied, in which case it is ignored.
wdist	The distribution of **W**. This can be one of "m" (multinomial), "p" (Poisson), "b" (binary) or "o" (other). If K.adj and K2 are given wdist is set to "o".
type	The type of saddlepoint approximation. Possible types are "simp" for simple saddlepoint and "cond" for the conditional saddlepoint. When wdist is "o" or "m", type is automatically set to "simp", which is the only type of saddlepoint currently implemented for those distributions.
d	This specifies the dimension of the whole statistic. This argument is required only when wdist = "o" and defaults to 1 if not supplied in that case. For other distributions it is set to ncol(A).
d1	When type is "cond" this is the dimension of the statistic of interest which must be less than length(u). Then the saddlepoint approximation to the conditional distribution of the first d1 linear combinations given the values of the remaining combinations is found. Conditional distribution function approximations can only be found if the value of d1 is 1.
init	Used if wdist is either "m" or "o", this gives initial values to nlmin which is used to solve the saddlepoint equation.
mu	The values of the parameters of the distribution of **W** when wdist is "m", "p" "b". mu must be of the same length as W (i.e. nrow(A)). The default is that all values of mu are equal and so the elements of **W** are identically distributed.

LR	If TRUE then the Lugananni-Rice approximation to the cdf is used, otherwise the approximation used is based on Barndorff-Nielsen's r*.
strata	The strata for stratified data.
K.adj	The adjusted cumulant generating function used when wdist is "o". This is a function of a single parameter, zeta, which calculates K(zeta)-u%*%zeta, where K(zeta) is the cumulant generating function of **W**.
K2	This is a function of a single parameter zeta which returns the matrix of second derivatives of K(zeta) for use when wdist is "o". If K.adj is given then this must be given also. It is called only once with the calculated solution to the saddlepoint equation being passed as the argument. This argument is ignored if K.adj is not supplied.

Details

If wdist is "o" or "m", the saddlepoint equations are solved using nlmin to minimize K.adj with respect to its parameter zeta. For the Poisson and binary cases, a generalized linear model is fitted such that the parameter estimates solve the saddlepoint equations. The response variable 'y' for the glm must satisfy the equation t(A)%*%y = u (t() being the transpose function). Such a vector can be found as a feasible solution to a linear programming problem. This is done by a call to simplex. The covariate matrix for the glm is given by A.

Value

A list consisting of the following components

spa	The saddlepoint approximations. The first value is the density approximation and the second value is the distribution function approximation.
zeta.hat	The solution to the saddlepoint equation. For the conditional saddlepoint this is the solution to the saddlepoint equation for the numerator.
zeta2.hat	If type is "cond" this is the solution to the saddlepoint equation for the denominator. This component is not returned for any other value of type.

References

Booth, J.G. and Butler, R.W. (1990) Randomization distributions and saddlepoint approximations in generalized linear models. *Biometrika*, **77**, 787–796.

Canty, A.J. and Davison, A.C. (1997) Implementation of saddlepoint approximations to resampling distributions. *Computing Science and Statistics; Proceedings of the 28th Symposium on the Interface*, 248–253.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press.

Jensen, J.L. (1995) *Saddlepoint Approximations*. Oxford University Press.

See Also

saddle.distn, simplex

Examples

```
# To evaluate the bootstrap distribution of the mean failure time of
# air-conditioning equipment at 80 hours
saddle(A = aircondit$hours/12, u = 80)

# Alternatively this can be done using a conditional poisson
saddle(A = cbind(aircondit$hours/12,1), u = c(80, 12),
       wdist = "p", type = "cond")

# To use the Lugananni-Rice approximation to this
saddle(A = cbind(aircondit$hours/12,1), u = c(80, 12),
       wdist = "p", type = "cond",
       LR = TRUE)

# Example 9.16 of Davison and Hinkley (1997) calculates saddlepoint
# approximations to the distribution of the ratio statistic for the
# city data. Since the statistic is not in itself a linear combination
# of random Variables, its distribution cannot be found directly.
# Instead the statistic is expressed as the solution to a linear
# estimating equation and hence its distribution can be found.  We
# get the saddlepoint approximation to the pdf and cdf evaluated at
# t = 1.25 as follows.
jacobian <- function(dat,t,zeta)
{
     p <- exp(zeta*(dat$x-t*dat$u))
     abs(sum(dat$u*p)/sum(p))
}
city.sp1 <- saddle(A = city$x-1.25*city$u, u = 0)
city.sp1$spa[1] <- jacobian(city, 1.25, city.sp1$zeta.hat) * city.sp1$spa[1]
city.sp1
```

---

saddle.distn *Saddlepoint Distribution Approximations for Bootstrap Statistics*

Description

Approximate an entire distribution using saddlepoint methods. This function can calculate simple and conditional saddlepoint distribution approximations for a univariate quantity of interest. For the simple saddlepoint the quantity of interest is a linear combination of **W** where **W** is a vector of random variables. For the conditional saddlepoint we require the distribution of one linear combination given the values of any number of other linear combinations. The distribution of **W** must be one of multinomial, Poisson or binary. The primary use of this function is to calculate quantiles of bootstrap distributions using saddlepoint approximations. Such quantiles are required by the function `control` to approximate the distribution of the linear approximation to a statistic.

Usage

```
saddle.distn(A, u = NULL, alpha = NULL, wdist = "m",
             type = "simp", npts = 20, t = NULL, t0 = NULL,
```

```
init = rep(0.1, d), mu = rep(0.5, n), LR = FALSE,
strata = NULL, ...)
```

Arguments

A	This is a matrix of known coefficients or a function which returns such a matrix. If a function then its first argument must be the point t at which a saddlepoint is required. The most common reason for A being a function would be if the statistic is not itself a linear combination of the **W** but is the solution to a linear estimating equation.
u	If A is a function then u must also be a function returning a vector with length equal to the number of columns of the matrix returned by A. Usually all components other than the first will be constants as the other components are the values of the conditioning variables. If A is a matrix with more than one column (such as when wdist = "cond") then u should be a vector with length one less than ncol(A). In this case u specifies the values of the conditioning variables. If A is a matrix with one column or a vector then u is not used.
alpha	The alpha levels for the quantiles of the distribution which should be returned. By default the 0.1, 0.5, 1, 2.5, 5, 10, 20, 50, 80, 90, 95, 97.5, 99, 99.5 and 99.9 percentiles are calculated.
wdist	The distribution of **W**. Possible values are "m" (multinomial), "p" (Poisson), or "b" (binary).
type	The type of saddlepoint to be used. Possible values are "simp" (simple saddlepoint) and "cond" (conditional). If wdist is "m", type is set to "simp".
npts	The number of points at which the saddlepoint approximation should be calculated and then used to fit the spline.
t	A vector of points at which the saddlepoint approximations are calculated. These points should extend beyond the extreme quantiles required but still be in the possible range of the bootstrap distribution. The observed value of the statistic should not be included in t as the distribution function approximation breaks down at that point. The points should, however cover the entire effective range of the distribution including close to the centre. If t is supplied then npts is set to length(t). When t is not supplied, the function attempts to find the effective range of the distribution and then selects points to cover this range.
t0	If t is not supplied then a vector of length 2 should be passed as t0. The first component of t0 should be the centre of the distribution and the second should be an estimate of spread (such as a standard error). These two are then used to find the effective range of the distribution. The range finding mechanism does rely on an accurate estimate of location in t0[1].
init	When wdist is "m", this vector should contain the initial values to be passed to nlmin when it is called to solve the saddlepoint equations.
mu	The vector of parameter values for the distribution. The default is that the components of **W** are identically distributed.
LR	A logical flag. When LR is TRUE the Lugananni-Rice cdf approximations are calculated and used to fit the spline. Otherwise the cdf approximations used are based on Barndorff-Nielsen's r*.

strata	A vector giving the strata when the rows of A relate to stratified data. This is used only when wdist is "m".
...	When A and u are functions any additional arguments are passed unchanged each time one of them is called.

Details

The range at which the saddlepoint is used is such that the cdf approximation at the endpoints is more extreme than required by the extreme values of alpha. The lower endpoint is found by evaluating the saddlepoint at the points t0[1]-2*t0[2], t0[1]-4*t0[2], t0[1]-8*t0[2] etc. until a point is found with a cdf approximation less than min(alpha)/10, then a bisection method is used to find the endpoint which has cdf approximation in the range (min(alpha)/1000, min(alpha)/10). Then a number of, equally spaced, points are chosen between the lower endpoint and t0[1] until a total of npts/2 approximations have been made. The remaining npts/2 points are chosen to the right of t0[1] in a similar manner. Any points which are very close to the centre of the distribution are then omitted as the cdf approximations are not reliable at the centre. A smoothing spline is then fitted to the probit of the saddlepoint distribution function approximations at the remaining points and the required quantiles are predicted from the spline.

Sometimes the function will terminate with the message "Unable to find range". There are two main reasons why this may occur. One is that the distribution is too discrete and/or the required quantiles too extreme, this can cause the function to be unable to find a point within the allowable range which is beyond the extreme quantiles. Another possibility is that the value of t0[2] is too small and so too many steps are required to find the range. The first problem cannot be solved except by asking for less extreme quantiles, although for very discrete distributions the approximations may not be very good. In the second case using a larger value of t0[2] will usually solve the problem.

Value

The returned value is an object of class "saddle.distn". See the help file for saddle.distn.object for a description of such an object.

References

Booth, J.G. and Butler, R.W. (1990) Randomization distributions and saddlepoint approximations in generalized linear models. *Biometrika*, **77**, 787–796.

Canty, A.J. and Davison, A.C. (1997) Implementation of saddlepoint approximations to resampling distributions. *Computing Science and Statistics; Proceedings of the 28th Symposium on the Interface* 248–253.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press.

Jensen, J.L. (1995) *Saddlepoint Approximations*. Oxford University Press.

See Also

lines.saddle.distn, saddle, saddle.distn.object, smooth.spline

Examples

```
#  The bootstrap distribution of the mean of the air-conditioning
#  failure data: fails to find value on R (and probably on S too)
air.t0 <- c(mean(aircondit$hours), sqrt(var(aircondit$hours)/12))
## Not run: saddle.distn(A = aircondit$hours/12, t0 = air.t0)

# alternatively using the conditional poisson
saddle.distn(A = cbind(aircondit$hours/12, 1), u = 12, wdist = "p",
             type = "cond", t0 = air.t0)

# Distribution of the ratio of a sample of size 10 from the bigcity
# data, taken from Example 9.16 of Davison and Hinkley (1997).
ratio <- function(d, w) sum(d$x *w)/sum(d$u * w)
city.v <- var.linear(empinf(data = city, statistic = ratio))
bigcity.t0 <- c(mean(bigcity$x)/mean(bigcity$u), sqrt(city.v))
Afn <- function(t, data) cbind(data$x - t*data$u, 1)
ufn <- function(t, data) c(0,10)
saddle.distn(A = Afn, u = ufn, wdist = "b", type = "cond",
             t0 = bigcity.t0, data = bigcity)

# From Example 9.16 of Davison and Hinkley (1997) again, we find the
# conditional distribution of the ratio given the sum of city$u.
Afn <- function(t, data) cbind(data$x-t*data$u, data$u, 1)
ufn <- function(t, data) c(0, sum(data$u), 10)
city.t0 <- c(mean(city$x)/mean(city$u), sqrt(city.v))
saddle.distn(A = Afn, u = ufn, wdist = "p", type = "cond", t0 = city.t0,
             data = city)
```

---

saddle.distn.object *Saddlepoint Distribution Approximation Objects*

Description

Class of objects that result from calculating saddlepoint distribution approximations by a call to `saddle.distn`.

Generation

This class of objects is returned from calls to the function `saddle.distn`.

Methods

The class `"saddle.distn"` has methods for the functions `lines` and `print`.

Structure

Objects of class `"saddle.distn"` are implemented as a list with the following components.

quantiles A matrix with 2 columns. The first column contains the probabilities `alpha` and the second column contains the estimated quantiles of the distribution at those probabilities derived from the spline.

points A matrix of evaluations of the saddlepoint approximation. The first column contains the values of `t` which were used, the second and third contain the density and cdf approximations at those points and the rest of the columns contain the solutions to the saddlepoint equations. When `type` is `"simp"`, there is only one of those. When `type` is `"cond"` there are `2*d-1` where `d` is the number of columns in `A` or the output of `A(t,...{})`. The first `d` of these correspond to the numerator and the remainder correspond to the denominator.

distn An object of class `smooth.spline`. This corresponds to the spline fitted to the saddlepoint cdf approximations in points in order to approximate the entire distribution. For the structure of the object see `smooth.spline`.

call The original call to `saddle.distn` which generated the object.

LR A logical variable indicating whether the Lugananni-Rice approximations were used.

See Also

`lines.saddle.distn`, `saddle.distn`, `print.saddle.distn`

`salinity` *Water Salinity and River Discharge*

Description

The `salinity` data frame has 28 rows and 4 columns.

Biweekly averages of the water salinity and river discharge in Pamlico Sound, North Carolina were recorded between the years 1972 and 1977. The data in this set consists only of those measurements in March, April and May.

Usage

```
salinity
```

Format

This data frame contains the following columns:

`sal` The average salinity of the water over two weeks.

`lag` The average salinity of the water lagged two weeks. Since only spring is used, the value of `lag` is not always equal to the previous value of `sal`.

`trend` A factor indicating in which of the 6 biweekly periods between March and May, the observations were taken. The levels of the factor are from 0 to 5 with 0 being the first two weeks in March.

`dis` The amount of river discharge during the two weeks for which `sal` is the average salinity.

Source

The data were obtained from

Ruppert, D. and Carroll, R.J. (1980) Trimmed least squares estimation in the linear model. *Journal of the American Statistical Association*, **75**, 828–838.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`simplex` *Simplex Method for Linear Programming Problems*

Description

This function will optimize the linear function `a%*%x` subject to the constraints `A1%*%x <= b1`, `A2%*%x >= b2`, `A3%*%x = b3` and `x >= 0`. Either maximization or minimization is possible but the default is minimization.

Usage

```
simplex(a, A1 = NULL, b1 = NULL, A2 = NULL, b2 = NULL, A3 = NULL,
        b3 = NULL, maxi = FALSE, n.iter = n + 2 * m, eps = 1e-10)
```

Arguments

`a`	A vector of length `n` which gives the coefficients of the objective function.
`A1`	An `m1` by `n` matrix of coefficients for the $\leq$ type of constraints.
`b1`	A vector of length `m1` giving the right hand side of the $\leq$ constraints. This argument is required if `A1` is given and ignored otherwise. All values in `b1` must be non-negative.
`A2`	An `m2` by `n` matrix of coefficients for the $\geq$ type of constraints.
`b2`	A vector of length `m2` giving the right hand side of the $\geq$ constraints. This argument is required if `A2` is given and ignored otherwise. All values in `b2` must be non-negative. Note that the constraints `x >= 0` are included automatically and so should not be repeated here.
`A3`	An `m3` by `n` matrix of coefficients for the equality constraints.
`b3`	A vector of length `m3` giving the right hand side of equality constraints. This argument is required if `A3` is given and ignored otherwise. All values in `b3` must be non-negative.
`maxi`	A logical flag which specifies minimization if `FALSE` (default) and maximization otherwise. If `maxi` is `TRUE` then the maximization problem is recast as a minimization problem by changing the objective function coefficients to their negatives.

`n.iter` The maximum number of iterations to be conducted in each phase of the simplex method. The default is `n+2*(m1+m2+m3)`.

`eps` The floating point tolerance to be used in tests of equality.

Details

The method employed by this function is the two phase tableau simplex method. If there are $\geq$ or equality constraints an initial feasible solution is not easy to find. To find a feasible solution an artificial variable is introduced into each $\geq$ or equality constraint and an auxiliary objective function is defined as the sum of these artificial variables. If a feasible solution to the set of constraints exists then the auxiliary objective will be minimized when all of the artificial variables are 0. These are then discarded and the original problem solved starting at the solution to the auxiliary problem. If the only constraints are of the $\leq$ form, the origin is a feasible solution and so the first stage can be omitted.

Value

An object of class `"simplex"`: see `simplex.object`.

Note

The method employed here is suitable only for relatively small systems. Also if possible the number of constraints should be reduced to a minimum in order to speed up the execution time which is approximately proportional to the cube of the number of constraints. In particular if there are any constraints of the form `x[i] >= b2[i]` they should be omitted by setting `x[i] = x[i]-b2[i]`, changing all the constraints and the objective function accordingly and then transforming back after the solution has been found.

References

Gill, P.E., Murray, W. and Wright, M.H. (1991) *Numerical Linear Algebra and Optimization Vol. 1*. Addison-Wesley.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992) *Numerical Recipes: The Art of Scientific Computing (Second Edition)*. Cambridge University Press.

Examples

```
# This example is taken from Exercise 7.5 of Gill, Murray and Wright (1991).
enj <- c(200, 6000, 3000, -200)
fat <- c(800, 6000, 1000, 400)
vitx <- c(50, 3, 150, 100)
vity <- c(10, 10, 75, 100)
vitz <- c(150, 35, 75, 5)
simplex(a = enj, A1 = fat, b1 = 13800, A2 = rbind(vitx, vity, vitz),
        b2 = c(600, 300, 550), maxi = TRUE)
```

simplex.object *Linear Programming Solution Objects*

Description

Class of objects that result from solving a linear programming problem using simplex.

Generation

This class of objects is returned from calls to the function simplex.

Methods

The class "saddle.distn" has a method for the function print.

Structure

Objects of class "simplex" are implemented as a list with the following components.

soln The values of x which optimize the objective function under the specified constraints provided those constraints are jointly feasible.

solved This indicates whether the problem was solved. A value of -1 indicates that no feasible solution could be found. A value of 0 that the maximum number of iterations was reached without termination of the second stage. This may indicate an unbounded function or simply that more iterations are needed. A value of 1 indicates that an optimal solution has been found.

value The value of the objective function at soln.

val.aux This is NULL if a feasible solution is found. Otherwise it is a positive value giving the value of the auxiliary objective function when it was minimized.

obj The original coefficients of the objective function.

a The objective function coefficients re-expressed such that the basic variables have coefficient zero.

a.aux This is NULL if a feasible solution is found. Otherwise it is the re-expressed auxiliary objective function at the termination of the first phase of the simplex method.

A The final constraint matrix which is expressed in terms of the non-basic variables. If a feasible solution is found then this will have dimensions m1+m2+m3 by n+m1+m2, where the final m1+m2 columns correspond to slack and surplus variables. If no feasible solution is found there will be an additional m1+m2+m3 columns for the artificial variables introduced to solve the first phase of the problem.

basic The indices of the basic (non-zero) variables in the solution. Indices between n+1 and n+m1 correspond to slack variables, those between n+m1+1 and n+m2 correspond to surplus variables and those greater than n+m2 are artificial variables. Indices greater than n+m2 should occur only if solved is -1 as the artificial variables are discarded in the second stage of the simplex method.

slack The final values of the m1 slack variables which arise when the "<=" constraints are re-expressed as the equalities A1%*%x + slack = b1.

surplus The final values of the m2 surplus variables which arise when the "<=" constraints are re-expressed as the equalities A2%*%x - surplus = b2.

artificial This is NULL if a feasible solution can be found. If no solution can be found then this contains the values of the m1+m2+m3 artificial variables which minimize their sum subject to the original constraints. A feasible solution exists only if all of the artificial variables can be made 0 simultaneously.

See Also

print.simplex, simplex

---

smooth.f *Smooth Distributions on Data Points*

Description

This function uses the method of frequency smoothing to find a distribution on a data set which has a required value, theta, of the statistic of interest. The method results in distributions which vary smoothly with theta.

Usage

```
smooth.f(theta, boot.out, index = 1, t = boot.out$t[, index],
         width = 0.5)
```

Arguments

theta	The required value for the statistic of interest. If theta is a vector, a separate distribution will be found for each element of theta.
boot.out	A bootstrap output object returned by a call to boot.
index	The index of the variable of interest in the output of boot.out$statistic. This argument is ignored if t is supplied. index must be a scalar.
t	The bootstrap values of the statistic of interest. This must be a vector of length boot.out$R and the values must be in the same order as the bootstrap replicates in boot.out.
width	The standardized width for the kernel smoothing. The smoothing uses a value of width*s for epsilon, where s is the bootstrap estimate of the standard error of the statistic of interest. width should take a value in the range (0.2, 1) to produce a reasonable smoothed distribution. If width is too large then the distribution becomes closer to uniform.

Details

The new distributional weights are found by applying a normal kernel smoother to the observed values of t weighted by the observed frequencies in the bootstrap simulation. The resulting distribution may not have parameter value exactly equal to the required value theta but it will typically have a value which is close to theta. The details of how this method works can be found in Davison, Hinkley and Worton (1995) and Section 3.9.2 of Davison and Hinkley (1997).

Value

If `length(theta)` is 1 then a vector with the same length as the data set `boot.out$data` is returned. The value in position `i` is the probability to be given to the data point in position `i` so that the distribution has parameter value approximately equal to `theta`. If `length(theta)` is bigger than 1 then the returned value is a matrix with `length(theta)` rows each of which corresponds to a distribution with the parameter value approximately equal to the corresponding value of `theta`.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Davison, A.C., Hinkley, D.V. and Worton, B.J. (1995) Accurate and efficient construction of bootstrap likelihoods. *Statistics and Computing*, **5**, 257–264.

See Also

`boot`, `exp.tilt`, `tilt.boot`

Examples

```
# Example 9.8 of Davison and Hinkley (1997) requires tilting the resampling
# distribution of the studentized statistic to be centred at the observed
# value of the test statistic 1.84.  In the book exponential tilting was used
# but it is also possible to use smooth.f.
grav1 <- gravity[as.numeric(gravity[, 2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2] - mns[1], s2hat, (mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
grav.boot <- boot(grav1, grav.fun, R = 499, stype = "w",
                  strata = grav1[, 2], orig = grav.z0[1])
grav.sm <- smooth.f(grav.z0[3], grav.boot, index = 3)

# Now we can run another bootstrap using these weights
grav.boot2 <- boot(grav1, grav.fun, R = 499, stype = "w",
                   strata = grav1[, 2], orig = grav.z0[1],
                   weights = grav.sm)

# Estimated p-values can be found from these as follows
mean(grav.boot$t[, 3] >= grav.z0[3])
imp.prob(grav.boot2, t0 = -grav.z0[3], t = -grav.boot2$t[, 3])


# Note that for the importance sampling probability we must
```

```
# multiply everything by -1 to ensure that we find the correct
# probability.  Raw resampling is not reliable for probabilities
# greater than 0.5. Thus
1 - imp.prob(grav.boot2, index = 3, t0 = grav.z0[3])$raw
# can give very strange results (negative probabilities).
```

`sunspot` *Annual Mean Sunspot Numbers*

Description

`sunspot` is a time series and contains 289 observations.

The Zurich sunspot numbers have been analyzed in almost all books on time series analysis as well as numerous papers. The data set, usually attributed to Rudolf Wolf, consists of means of daily relative numbers of sunspot sightings. The relative number for a day is given by k(f+10g) where g is the number of sunspot groups observed, f is the total number of spots within the groups and k is a scaling factor relating the observer and telescope to a baseline. The relative numbers are then averaged to give an annual figure. See Inzenman (1983) for a discussion of the relative numbers. The figures are for the years 1700-1988.

Source

The data were obtained from

Tong, H. (1990) *Nonlinear Time Series: A Dynamical System Approach*. Oxford University Press

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Inzenman, A.J. (1983) J.R. Wolf and H.A. Wolfer: An historical note on the Zurich sunspot relative numbers. *Journal of the Royal Statistical Society, A*, **146**, 311-318.

Waldmeir, M. (1961) *The Sunspot Activity in the Years 1610-1960*. Schulthess and Co.

`survival` *Survival of Rats after Radiation Doses*

Description

The `survival` data frame has 14 rows and 2 columns.

The data measured the survival percentages of batches of rats who were given varying doses of radiation. At each of 6 doses there were two or three replications of the experiment.

Usage

```
survival
```

Format

This data frame contains the following columns:

`dose` The dose of radiation administered (rads).

`surv` The survival rate of the batches expressed as a percentage.

Source

The data were obtained from

Efron, B. (1988) Computer-intensive methods in statistical regression. *SIAM Review*, **30**, 421–449.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

---

tau *Tau Particle Decay Modes*

Description

The `tau` data frame has 60 rows and 2 columns.

The tau particle is a heavy electron-like particle discovered in the 1970's by Martin Perl at the Stanford Linear Accelerator Center. Soon after its production the tau particle decays into various collections of more stable particles. About 86% of the time the decay involves just one charged particle. This rate has been measured independently 13 times.

The one-charged-particle event is made up of four major modes of decay as well as a collection of other events. The four main types of decay are denoted rho, pi, e and mu. These rates have been measured independently 6, 7, 14 and 19 times respectively. Due to physical constraints each experiment can only estimate the composite one-charged-particle decay rate or the rate of one of the major modes of decay.

Each experiment consists of a major research project involving many years work. One of the goals of the experiments was to estimate the rate of decay due to events other than the four main modes of decay. These are uncertain events and so cannot themselves be observed directly.

Usage

```
tau
```

Format

This data frame contains the following columns:

`rate` The decay rate expressed as a percentage.

`decay` The type of decay measured in the experiment. It is a factor with levels `1`, `rho`, `pi`, `e` and `mu`.

Source

The data were obtained from

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions (with Discussion). *Journal of the Royal Statistical Society, B*, **54**, 83–127.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Hayes, K.G., Perl, M.L. and Efron, B. (1989) Application of the bootstrap statistical method to the tau-decay-mode problem. *Physical Review, D*, **39**, 274-279.

---

tilt.boot *Non-parametric Tilted Bootstrap*

Description

This function will run an initial bootstrap with equal resampling probabilities (if required) and will use the output of the initial run to find resampling probabilities which put the value of the statistic at required values. It then runs an importance resampling bootstrap using the calculated probabilities as the resampling distribution.

Usage

```
tilt.boot(data, statistic, R, sim = "ordinary", stype = "i",
          strata = rep(1, n), L = NULL, theta = NULL,
          alpha = c(0.025, 0.975), tilt = TRUE, width = 0.5,
          index = 1, ...)
```

Arguments

`data`	The data as a vector, matrix or data frame. If it is a matrix or data frame then each row is considered as one (multivariate) observation.
`statistic`	A function which when applied to data returns a vector containing the statistic(s) of interest. It must take at least two arguments. The first argument will always be `data` and the second should be a vector of indices, weights or frequencies describing the bootstrap sample. Any other arguments must be supplied to `tilt.boot` and will be passed unchanged to statistic each time it is called.
`R`	The number of bootstrap replicates required. This will generally be a vector, the first value stating how many uniform bootstrap simulations are to be performed at the initial stage. The remaining values of `R` are the number of simulations to be performed resampling from each reweighted distribution. The first value of `R` must always be present, a value of 0 implying that no uniform resampling is to be carried out. Thus `length(R)` should always equal `1+length(theta)`.

sim	This is a character string indicating the type of bootstrap simulation required. There are only two possible values that this can take: "ordinary" and "balanced". If other simulation types are required for the initial un-weighted bootstrap then it will be necessary to run boot, calculate the weights appropriately, and run boot again using the calculated weights.
stype	A character string indicating the type of second argument expected by statistic. The possible values that stype can take are "i" (indices), "w" (weights) and "f" (frequencies).
strata	An integer vector or factor representing the strata for multi-sample problems.
L	The empirical influence values for the statistic of interest. They are used only for exponential tilting when tilt is TRUE. If tilt is TRUE and they are not supplied then tilt.boot uses empinf to calculate them.
theta	The required parameter value(s) for the tilted distribution(s). There should be one value of theta for each of the non-uniform distributions. If R[1] is 0 theta is a required argument. Otherwise theta values can be estimated from the initial uniform bootstrap and the values in alpha.
alpha	The alpha level to which tilting is required. This parameter is ignored if R[1] is 0 or if theta is supplied, otherwise it is used to find the values of theta as quantiles of the initial uniform bootstrap. In this case R[1] should be large enough that min(c(alpha, 1-alpha))*R[1] > 5, if this is not the case then a warning is generated to the effect that the theta are extreme values and so the tilted output may be unreliable.
tilt	A logical variable which if TRUE (the default) indicates that exponential tilting should be used, otherwise local frequency smoothing (smooth.f) is used. If tilt is FALSE then R[1] must be positive. In fact in this case the value of R[1] should be fairly large (in the region of 500 or more).
width	This argument is used only if tilt is FALSE, in which case it is passed unchanged to smooth.f as the standardized bandwidth for the smoothing operation. The value should generally be in the range (0.2, 1). See smooth.f for for more details.
index	The index of the statistic of interest in the output from statistic. By default the first element of the output of statistic is used.
...	Any additional arguments required by statistic. These are passed unchanged to statistic each time it is called.

Value

An object of class "boot" with the following components

t0	The observed value of the statistic on the original data.
t	The values of the bootstrap replicates of the statistic. There will be sum(R) of these, the first R[1] corresponding to the uniform bootstrap and the remainder to the tilted bootstrap(s).
R	The input vector of the number of bootstrap replicates.
data	The original data as supplied.

statistic	The statistic function as supplied.
sim	The simulation type used in the bootstrap(s), it can either be "ordinary" or "balanced".
stype	The type of statistic supplied, it is the same as the input value stype.
call	A copy of the original call to tilt.boot.
strata	The strata as supplied.
weights	The matrix of weights used. If R[1] is greater than 0 then the first row will be the uniform weights and each subsequent row the tilted weights. If R[1] equals 0 then the uniform weights are omitted and only the tilted weights are output.
theta	The values of theta used for the tilted distributions. These are either the input values or the values derived from the uniform bootstrap and alpha.

References

Booth, J.G., Hall, P. and Wood, A.T.A. (1993) Balanced importance resampling for the bootstrap. *Annals of Statistics*, **21**, 286–298.

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Hinkley, D.V. and Shi, S. (1989) Importance sampling and the nested bootstrap. *Biometrika*, **76**, 435–446.

See Also

boot, exp.tilt, Imp.Estimates, imp.weights, smooth.f

Examples

```
# Note that these examples can take a while to run.

# Example 9.9 of Davison and Hinkley (1997).
grav1 <- gravity[as.numeric(gravity[,2]) >= 7, ]
grav.fun <- function(dat, w, orig) {
     strata <- tapply(dat[, 2], as.numeric(dat[, 2]))
     d <- dat[, 1]
     ns <- tabulate(strata)
     w <- w/tapply(w, strata, sum)[strata]
     mns <- as.vector(tapply(d * w, strata, sum)) # drop names
     mn2 <- tapply(d * d * w, strata, sum)
     s2hat <- sum((mn2 - mns^2)/ns)
     c(mns[2]-mns[1],s2hat,(mns[2]-mns[1]-orig)/sqrt(s2hat))
}
grav.z0 <- grav.fun(grav1, rep(1, 26), 0)
tilt.boot(grav1, grav.fun, R = c(249, 375, 375), stype = "w",
          strata = grav1[,2], tilt = TRUE, index = 3, orig = grav.z0[1])


#  Example 9.10 of Davison and Hinkley (1997) requires a balanced
#  importance resampling bootstrap to be run.  In this example we
#  show how this might be run.
```

```
acme.fun <- function(data, i, bhat) {
     d <- data[i,]
     n <- nrow(d)
     d.lm <- glm(d$acme~d$market)
     beta.b <- coef(d.lm)[2]
     d.diag <- boot::glm.diag(d.lm)
     SSx <- (n-1)*var(d$market)
     tmp <- (d$market-mean(d$market))*d.diag$res*d.diag$sd
     sr <- sqrt(sum(tmp^2))/SSx
     c(beta.b, sr, (beta.b-bhat)/sr)
}
acme.b <- acme.fun(acme, 1:nrow(acme), 0)
acme.boot1 <- tilt.boot(acme, acme.fun, R = c(499, 250, 250),
                        stype = "i", sim = "balanced", alpha = c(0.05, 0.95),
                        tilt = TRUE, index = 3, bhat = acme.b[1])
```

`tsboot` *Bootstrapping of Time Series*

Description

Generate `R` bootstrap replicates of a statistic applied to a time series. The replicate time series can be generated using fixed or random block lengths or can be model based replicates.

Usage

```
tsboot(tseries, statistic, R, l = NULL, sim = "model",
       endcorr = TRUE, n.sim = NROW(tseries), orig.t = TRUE,
       ran.gen, ran.args = NULL, norm = TRUE, ...,
       parallel = c("no", "multicore", "snow"),
       ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

Arguments

`tseries`	A univariate or multivariate time series.
`statistic`	A function which when applied to `tseries` returns a vector containing the statistic(s) of interest. Each time `statistic` is called it is passed a time series of length `n.sim` which is of the same class as the original `tseries`. Any other arguments which `statistic` takes must remain constant for each bootstrap replicate and should be supplied through the ... argument to `tsboot`.
`R`	A positive integer giving the number of bootstrap replicates required.
`sim`	The type of simulation required to generate the replicate time series. The possible input values are `"model"` (model based resampling), `"fixed"` (block resampling with fixed block lengths of `l`), `"geom"` (block resampling with block lengths having a geometric distribution with mean `l`) or `"scramble"` (phase scrambling).

`l` If `sim` is `"fixed"` then `l` is the fixed block length used in generating the replicate time series. If `sim` is `"geom"` then `l` is the mean of the geometric distribution used to generate the block lengths. `l` should be a positive integer less than the length of `tseries`. This argument is not required when `sim` is `"model"` but it is required for all other simulation types.

`endcorr` A logical variable indicating whether end corrections are to be applied when `sim` is `"fixed"`. When `sim` is `"geom"`, `endcorr` is automatically set to `TRUE`; `endcorr` is not used when `sim` is `"model"` or `"scramble"`.

`n.sim` The length of the simulated time series. Typically this will be equal to the length of the original time series but there are situations when it will be larger. One obvious situation is if prediction is required. Another situation in which `n.sim` is larger than the original length is if `tseries` is a residual time series from fitting some model to the original time series. In this case, `n.sim` would usually be the length of the original time series.

`orig.t` A logical variable which indicates whether `statistic` should be applied to `tseries` itself as well as the bootstrap replicate series. If `statistic` is expecting a longer time series than `tseries` or if applying `statistic` to `tseries` will not yield any useful information then `orig.t` should be set to `FALSE`.

`ran.gen` This is a function of three arguments. The first argument is a time series. If `sim` is `"model"` then it will always be `tseries` that is passed. For other simulation types it is the result of selecting `n.sim` observations from `tseries` by some scheme and converting the result back into a time series of the same form as `tseries` (although of length `n.sim`). The second argument to `ran.gen` is always the value `n.sim`, and the third argument is `ran.args`, which is used to supply any other objects needed by `ran.gen`. If `sim` is `"model"` then the generation of the replicate time series will be done in `ran.gen` (for example through use of `arima.sim`). For the other simulation types `ran.gen` is used for 'post-blackening'. The default is that the function simply returns the time series passed to it.

`ran.args` This will be supplied to `ran.gen` each time it is called. If `ran.gen` needs any extra arguments then they should be supplied as components of `ran.args`. Multiple arguments may be passed by making `ran.args` a list. If `ran.args` is `NULL` then it should not be used within `ran.gen` but note that `ran.gen` must still have its third argument.

`norm` A logical argument indicating whether normal margins should be used for phase scrambling. If `norm` is `FALSE` then margins corresponding to the exact empirical margins are used.

`...` Extra named arguments to `statistic` may be supplied here. Beware of partial matching to the arguments of `tsboot` listed above.

`parallel, ncpus, cl`
See the help for `boot`.

Details

If `sim` is `"fixed"` then each replicate time series is found by taking blocks of length `l`, from the original time series and putting them end-to-end until a new series of length `n.sim` is created. When

sim is "geom" a similar approach is taken except that now the block lengths are generated from a geometric distribution with mean l. Post-blackening can be carried out on these replicate time series by including the function ran.gen in the call to tsboot and having tseries as a time series of residuals.

Model based resampling is very similar to the parametric bootstrap and all simulation must be in one of the user specified functions. This avoids the complicated problem of choosing the block length but relies on an accurate model choice being made.

Phase scrambling is described in Section 8.2.4 of Davison and Hinkley (1997). The types of statistic for which this method produces reasonable results is very limited and the other methods seem to do better in most situations. Other types of resampling in the frequency domain can be accomplished using the function boot with the argument sim = "parametric".

Value

An object of class "boot" with the following components.

t0	If orig.t is TRUE then t0 is the result of statistic(tseries,...{}) otherwise it is NULL.
t	The results of applying statistic to the replicate time series.
R	The value of R as supplied to tsboot.
tseries	The original time series.
statistic	The function statistic as supplied.
sim	The simulation type used in generating the replicates.
endcorr	The value of endcorr used. The value is meaningful only when sim is "fixed"; it is ignored for model based simulation or phase scrambling and is always set to TRUE if sim is "geom".
n.sim	The value of n.sim used.
l	The value of l used for block based resampling. This will be NULL if block based resampling was not used.
ran.gen	The ran.gen function used for generating the series or for 'post-blackening'.
ran.args	The extra arguments passed to ran.gen.
call	The original call to tsboot.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Kunsch, H.R. (1989) The jackknife and the bootstrap for general stationary observations. *Annals of Statistics*, **17**, 1217–1241.

Politis, D.N. and Romano, J.P. (1994) The stationary bootstrap. *Journal of the American Statistical Association*, **89**, 1303–1313.

See Also

boot, arima.sim

Examples

```
lynx.fun <- function(tsb) {
     ar.fit <- ar(tsb, order.max = 25)
     c(ar.fit$order, mean(tsb), tsb)
}

# the stationary bootstrap with mean block length 20
lynx.1 <- tsboot(log(lynx), lynx.fun, R = 99, l = 20, sim = "geom")

# the fixed block bootstrap with length 20
lynx.2 <- tsboot(log(lynx), lynx.fun, R = 99, l = 20, sim = "fixed")

# Now for model based resampling we need the original model
# Note that for all of the bootstraps which use the residuals as their
# data, we set orig.t to FALSE since the function applied to the residual
# time series will be meaningless.
lynx.ar <- ar(log(lynx))
lynx.model <- list(order = c(lynx.ar$order, 0, 0), ar = lynx.ar$ar)
lynx.res <- lynx.ar$resid[!is.na(lynx.ar$resid)]
lynx.res <- lynx.res - mean(lynx.res)

lynx.sim <- function(res,n.sim, ran.args) {
     # random generation of replicate series using arima.sim
     rg1 <- function(n, res) sample(res, n, replace = TRUE)
     ts.orig <- ran.args$ts
     ts.mod <- ran.args$model
     mean(ts.orig)+ts(arima.sim(model = ts.mod, n = n.sim,
                      rand.gen = rg1, res = as.vector(res)))
}

lynx.3 <- tsboot(lynx.res, lynx.fun, R = 99, sim = "model", n.sim = 114,
                 orig.t = FALSE, ran.gen = lynx.sim,
                 ran.args = list(ts = log(lynx), model = lynx.model))

#  For "post-blackening" we need to define another function
lynx.black <- function(res, n.sim, ran.args) {
     ts.orig <- ran.args$ts
     ts.mod <- ran.args$model
     mean(ts.orig) + ts(arima.sim(model = ts.mod,n = n.sim,innov = res))
}

# Now we can run apply the two types of block resampling again but this
# time applying post-blackening.
lynx.1b <- tsboot(lynx.res, lynx.fun, R = 99, l = 20, sim = "fixed",
                  n.sim = 114, orig.t = FALSE, ran.gen = lynx.black,
                  ran.args = list(ts = log(lynx), model = lynx.model))

lynx.2b <- tsboot(lynx.res, lynx.fun, R = 99, l = 20, sim = "geom",
                  n.sim = 114, orig.t = FALSE, ran.gen = lynx.black,
                  ran.args = list(ts = log(lynx), model = lynx.model))

# To compare the observed order of the bootstrap replicates we
```

```
# proceed as follows.
table(lynx.1$t[, 1])
table(lynx.1b$t[, 1])
table(lynx.2$t[, 1])
table(lynx.2b$t[, 1])
table(lynx.3$t[, 1])
# Notice that the post-blackened and model-based bootstraps preserve
# the true order of the model (11) in many more cases than the others.
```

---

tuna *Tuna Sighting Data*

Description

The tuna data frame has 64 rows and 1 columns.

The data come from an aerial line transect survey of Southern Bluefin Tuna in the Great Australian Bight. An aircraft with two spotters on board flies randomly allocated line transects. Each school of tuna sighted is counted and its perpendicular distance from the transect measured. The survey was conducted in summer when tuna tend to stay on the surface.

Usage

```
tuna
```

Format

This data frame contains the following column:

y The perpendicular distance, in miles, from the transect for 64 independent sightings of tuna schools.

Source

The data were obtained from

Chen, S.X. (1996) Empirical likelihood confidence intervals for nonparametric density estimation. *Biometrika*, **83**, 329–341.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`urine` *Urine Analysis Data*

Description

The `urine` data frame has 79 rows and 7 columns.

79 urine specimens were analyzed in an effort to determine if certain physical characteristics of the urine might be related to the formation of calcium oxalate crystals.

Usage

```
urine
```

Format

This data frame contains the following columns:

`r` Indicator of the presence of calcium oxalate crystals.

`gravity` The specific gravity of the urine.

`ph` The pH reading of the urine.

`osmo` The osmolarity of the urine. Osmolarity is proportional to the concentration of molecules in solution.

`cond` The conductivity of the urine. Conductivity is proportional to the concentration of charged ions in solution.

`urea` The urea concentration in millimoles per litre.

`calc` The calcium concentration in millimoles per litre.

Source

The data were obtained from

Andrews, D.F. and Herzberg, A.M. (1985) *Data: A Collection of Problems from Many Fields for the Student and Research Worker*. Springer-Verlag.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

`var.linear` *Linear Variance Estimate*

Description

Estimates the variance of a statistic from its empirical influence values.

Usage

```
var.linear(L, strata = NULL)
```

Arguments

`L`	Vector of the empirical influence values of a statistic. These will usually be calculated by a call to `empinf`.
`strata`	A numeric vector or factor specifying which observations (and hence empirical influence values) come from which strata.

Value

The variance estimate calculated from `L`.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

See Also

`empinf`, `linear.approx`, `k3.linear`

Examples

```
#  To estimate the variance of the ratio of means for the city data.
ratio <- function(d,w) sum(d$x * w)/sum(d$u * w)
var.linear(empinf(data = city, statistic = ratio))
```

wool *Australian Relative Wool Prices*

Description

`wool` is a time series of class `"ts"` and contains 309 observations.

Each week that the market is open the Australian Wool Corporation set a floor price which determines their policy on intervention and is therefore a reflection of the overall price of wool for the week in question. Actual prices paid can vary considerably about the floor price. The series here is the log of the ratio between the price for fine grade wool and the floor price, each market week between July 1976 and Jun 1984.

Source

The data were obtained from

Diggle, P.J. (1990) *Time Series: A Biostatistical Introduction*. Oxford University Press.

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application*. Cambridge University Press.

Index