

Package ‘cir’

April 27, 2023

Type Package

Title Centered Isotonic Regression and Dose-Response Utilities

Version 2.3.1

Date 2023-04-26

Description Isotonic regression (IR) and its improvement: centered isotonic regression (CIR). CIR is recommended in particular with small samples. Also, interval estimates for both, and additional utilities such as plotting dose-response data. For dev version and change history, see GitHub [assaforon/cir](#).

License GPL-2

RoxygenNote 7.2.3

Encoding UTF-8

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation no

Author Assaf P. Oron [cre, aut]

Maintainer Assaf P. Oron <assaf.oron@gmail.com>

Repository CRAN

Date/Publication 2023-04-27 08:22:41 UTC

R topics documented:

cir-package	2
cirPAVA	3
deltaInverse	5
doseFind	8
DRshrink	10
is.DRtrace	11
isotInterval	12
iterCIR	15
morrisCI	16
oldPAVA	18

plot.DRtrace	19
quickInverse	22
quickIsotone	24
slope	27
wilsonCI	28

Index	30
--------------	-----------

cir-package	<i>Isotonic Regression, Centered Isotonic Regression, and Dose-Response Utilities</i>
-------------	---

Description

This package revolves around centered isotonic regression (CIR), an improvement to isotonic regression (IR). However, it also includes a flexible, useful implementation of IR, confidence-interval estimates for both CIR and IR, and additional utilities for dose-response and dose-finding data.

Details

Isotonic regression (IR) is a standard nonparametric estimation method for monotone data. We have developed an improvement to univariate IR, named centered isotonic regression (CIR). There are heuristic and theoretical justifications to prefer CIR over IR, but first and foremost, in most simulations it produces substantially smaller estimation error. More details appear in Oron and Flournoy (2017).

This package implements CIR, but "along the way" an enhanced interface to univariate IR is also available. IR's base-R implementation `isoreg` is very limited, as its own help page admits. A few other packages provide versions of IR, but to my knowledge the `cir` implementation offers some unique conveniences.

In addition, Oron and Flournoy (2017) also develop theoretically-backed confidence intervals applicable to both CIR and IR. The package's convenience wrapper `quickIsotone` executes CIR (or IR if one chooses `estfun = oldPAVA`), and returns both point and interval estimates at the specified x values.

Since our motivation for studying IR comes from dose-finding designs such as Up-and-Down, there's analogous functionality for dose-finding ("inverse") estimation of x given specified y values. In particular, `quickInverse` offers inverse point and interval estimates in a single call. The package now also includes an optional bias-correction shrinkage method for such designs, informed by more recent research (Flournoy and Oron, 2020).

The package's focus is dose-response data with the response assumed binary (coded as 0 or 1). Some functions might work for any input data, but others will not. In particular, the confidence intervals are only applicable to binary-response data.

The package also includes two S3 classes, `doseResponse` and `DRtrace`. The former which is more heavily used, is a data frame with elements x, y, wt , summarizing the dose-response information. The latter is a "trace" or a running description of raw dose-response data, with x, y provided at the resolution of single observations. Each class has a `plot` method.

Enjoy!

Author(s)

Assaf P. Oron.

Maintainer: Assaf P. Oron <assaf.oron.at.gmail.com>

References

Oron, A.P. and Flournoy, N., 2017. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. *Statistics in Biopharmaceutical Research* 9, 258-267. (author's public version available on arxiv.org).

Flournoy, N. and Oron, A.P., 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

cirPAVA

Returns centered-isotonic-regression estimate

Description

Nonparametric forward point estimation of a monotone response (y) as a function of dose (x), using the centered-isotonic-regression (CIR) algorithm.

Usage

```

cirPAVA(
  y,
  x = NULL,
  wt = NULL,
  outx = NULL,
  full = FALSE,
  dec = FALSE,
  strict = FALSE,
  interiorStrict = TRUE,
  ybounds = 0:1,
  adaptiveShrink = FALSE,
  ...
)

```

Arguments

y	can be either of the following: y values (response rates), a DRtrace object, a doseResponse object, or valid input (potentially together with x, wt) to generate a doseResponse object. See doseResponse help for more.
x	dose levels (if not included in y).
wt	weights (if not included in y).
outx	vector of x values for which predictions will be made. If NULL (default), this will be set to the set of unique values in the x argument (or equivalently in y\$x). Non-NULL inputs are relevant only if full=TRUE.

full	logical, is a more complete output desired? if FALSE (default), only a vector of point estimates for y at the provided dose levels is returned
dec	logical, is the true function is assumed to be monotone decreasing? Default FALSE.
strict	logical, should CIR enforce strict monotonicity by "fixing" flat intervals as well? Default FALSE.
interiorStrict	logical, should CIR enforce strict monotonicity, but only for y values inside of ybounds? Default TRUE. Choosing FALSE will be overridden if strict=TRUE, and a warning will be given.
ybounds	numeric vector of length 2, relevant only under the default setting of strict=FALSE, interiorStrict=TRUE. Default 0:1. See 'Details'.
adaptiveShrink	logical, should the y-values be pre-shrunk towards an experiment's target? Recommended if data were obtained via an adaptive dose-finding design. If TRUE, then must also provide a target argument that will be passed via
. . .	Other arguments passed on to pre-processing functions.

Details

This is the underlying "engine" function implementing CIR. For a quick and somewhat more user-friendly wrapper, use [quickIsotone](#). CIR is a variation of isotonic regression (IR) that shrinks IR's constant ("flat") intervals to single points and interpolates between these points, generating a curve that is strictly monotone everywhere except (possibly) near the boundaries.

Flat intervals in the raw input data, are handled with care. Under the default setting (strict=FALSE, interiorStrict=TRUE), flat intervals are treated as monotonicity violations, unless the y value is on the boundary of its allowed range (default $[0, 1]$, appropriate for binary-response data). On that boundary, flat intervals are left unchanged.

The algorithm is documented and discussed in Oron and Flournoy (2017). The function now include an `adaptiveShrink` option, to mitigate bias caused when using adaptive designs (Flournoy and Oron, 2020).

Value

under default, returns a vector of y estimates at unique x values. With `full=TRUE`, returns a list of 3 [doseResponse](#) objects name `output`, `input`, `shrinkage` for the output data at dose levels, the input data, and the function as fit at algorithm-generated shrinkage points, respectively.

Author(s)

Assaf P. Oron <assaf.oron.at@gmail.com>

References

- Oron, A.P. and Flournoy, N., 2017. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. *Statistics in Biopharmaceutical Research* 9, 258-267. (author's public version available on [arxiv.org](#)).
- Flournoy, N. and Oron, A.P., 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

See Also

[oldPAVA](#), [quickIsotone](#); [DRshrink](#) for explanation about `adaptiveShrink`.

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# CIR, using the default 'quick' function that also provides CIs (default 90%).
# The experiment's goal is to find the 30th percentile. We deploy the empirical bias correction.
quick1=quickIsotone(dat, adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
quick1
# Use 'estfun' argument to operate the same function with old PAVA as the estimator
# Here we neglect the bias correction to sharpen the old:new contrast
quick0=quickIsotone(dat,estfun=oldPAVA)
quick0

### Showing the data and the fits
par(mar=c(3,3,1,1),mgp=c(2,.5,0),tcl=-0.25)
plot(dat, ylim=c(0.05,0.55), las=1) # uses plot.doseResponse()
# The IR fit: a straightforward interpolation
lines(quick0$y, lty=2)

# With CIR, 'quickIsotone' cannot show us the true underlying interpolation;
# it only provides the estimates at requested points. Interpolation should be done between
# shrinkage points, not the original design points. So we must call the full 'cirPAVA' function:

slow1 = cirPAVA(dat, full=TRUE, adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
# Now, compare these 3 (the first one is wrong, b/c it interpolates from design points):
midpts = 1:4 + 0.5
approx(1:5,quick1$y, xout=midpts)$y
# instead, you can just call 'quickIsotone' and specify 'outx'
quickIsotone(dat,outx=midpts , adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
approx(slow1$shrinkage$x,slow1$shrinkage$y,xout=midpts)$y # Or use 'cirPAVA'

# Ok... finally plotting the CIR curve
# Both flat intervals are shrunk, because neither are at y=0 or y=1
lines(slow1$shrinkage$x,slow1$shrinkage$y, lwd = 2)

# Last but not least, here's the true response function
lines(seq(1,5,0.1),pweibull(seq(1,5,0.1),shape=1.1615,scale=8.4839),col=2)
legend('topleft',pch=c(NA,'X',NA,NA),lty=c(1,NA,2,1),col=c(2,1,1,1),
legend=c('True Curve','Observations','IR','CIR'), bty='n')
```

Description

Calculate left-bound to right-bound intervals for the dose point estimates, using local slopes at design points (places where observations exist) to invert the forward lower-upper bounds.

Usage

```
deltaInverse(
  isotPoint,
  target = (1:3)/4,
  intfun = morrisCI,
  conf = 0.9,
  adaptiveCurve = FALSE,
  minslope = 0.01,
  slopeRefinement = TRUE,
  finegrid = 0.05,
  ...
)
```

Arguments

isotPoint	The output of an estimation function such as cirPAVA , doseFind , with the option <code>full=TRUE</code> . Should be at least a list of 3 doseResponse objects named <code>input</code> , <code>output</code> , <code>shrinkage</code> .
target	A vector of target response rate(s), for which the interval is needed. Default (since version 2.3.0) is the 3 quartiles $((1:3) / 4)$. If changed to <code>NULL</code> , interval will be returned for the y values of <code>isotPoint\$output</code> .
intfun	the function to be used for initial (forward) interval estimation. Default morrisCI (see help on that function for additional options).
conf	numeric, the interval's confidence level as a fraction in $(0,1)$. Default 0.9.
adaptiveCurve	logical, should the CIs be expanded by using a parabolic curve between estimation points rather than straight interpolation (default <code>FALSE</code>)? Recommended when adaptive design was used and target is not 0.5.
minslope	minimum local slope (subsequently normalized by the units dose spacing) considered positive, passed on to slope . Needed to avoid unrealistically broad intervals. Default 0.01.
slopeRefinement	(new to 2.3.0) logical: whether to allow refinement of the slope estimate, including different slopes to the left and right of target. Default <code>TRUE</code> . See Details.
finegrid	a numerical value used to guide how fine the grid of x values will be during slope estimation. Should be in $(0,1)$ (preferably much less than 1). Default 0.05.
...	additional arguments passed on to quickIsotone

Details

The Delta method in this application boils down to dividing the distance to the forward (vertical) bounds, by the slope, to get the left/right interval width. Both forward intervals and slopes are calculated across a standard set of x values, then interpolated at horizontal cross-sections determined by target. Slope estimates are performed by [slope](#).

Starting version 2.3.0, by default the slope estimate is different to the right and left of target. The intervals should now better accommodate the sharp slope changes that often happen with discrete dose-response datasets. Operationally, the intervals are first estimated via the single-slope approach described above. Then using a finer grid of x values, weighted-average slopes to the right and left of the point estimate separately are calculated over the first-stage's half-intervals. The weights are hard-coded as quadratic (Epanechnikov).

An alternative and much simpler interval method (dubbed "global") is hard-coded into `quickInverse`, and can be chosen from there as an option. But it is not recommended.

Value

two-column matrix with the left and right bounds, respectively

See Also

`quickIsotone`, `quickInverse`, `isotInterval`, `slope`; `DRshrink` for the shrinkage fix.

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17), wt=c(7,24,20,12,17))
# The experiment's goal is to find the 30th percentile
quick1=quickIsotone(dat, adaptiveShrink = TRUE, target = 0.3)

# For inverse confidence intervals "the long way",
# we need a full CIR output object:
fwd1=cirPAVA(dat, full=TRUE, adaptiveShrink = TRUE, target = 0.3)
# Inverse intervals.
# Note: 'target' specifies the y values at which the interval is calculated.
# They are selected here based on the y range in which there are estimates.
yvals = c(seq(0.15, 0.3, 0.05), 0.33)
invDelta=deltaInverse(fwd1, target = yvals, adaptiveCurve = TRUE)
# We added the adaptiveCurve option because the experiment's target is off-center,
# and inverse-interval coverage tends to be lacking w/o that option.

### Showing the data and the estimates
par(mar=c(3,3,4,1), mgp=c(2, .5, 0), tcl=-0.25)
# Following command uses plot.doseResponse()
plot(dat, ylim=c(0.05,0.55),
      las=1, xlim=c(0,6.5), main="Inverse-Estimation CIs")

# The true response function; true target is where it crosses the y=0.3 line
lines(seq(0,7,0.1), pweibull(seq(0,7,0.1), shape=1.1615, scale=8.4839), col=4, lwd=1.5)
abline(h=0.3, col=2, lwd=2, lty=3) ### The experiment's official target

# Forward CIs; the "global" inverse interval just draws horizontal lines between them
# To get these "global" intervals calculated for you at specific targets, choose 'delta=FALSE'
# when calling quickInverse()
lines(quick1$y, lwd=1.5, col='purple')
lines(quick1$lower90conf, lty=2, col=3)
lines(quick1$upper90conf, lty=2, col=3)
```

```
# Note that only the upper forward bounds intersect the horizontal line at y=0.3.
# Therefore, via the "global" approach there won't be a finite CI for the target estimate.

# Now, the default "local" inverse interval, which is finite for the range of estimated y values.
# In particular, it is finite (albeit very wide) for y=0.3.
lines(invDelta[,1], yvals, lty=2, lwd=2)
lines(invDelta[,2], yvals, lty=2, lwd=2)

legend('topleft', pch=c(NA,NA,'X',NA,NA), lty=c(1,1,NA,2,2),
       col=c(4,'purple', 1,1,3), lwd=c(1.5,1.5,0,2,1),
       legend = c('True Curve', 'CIR Curve', 'Observations',
                 'Local Interval (default)',
                 'Forward/Global Interval'), bty='n')
```

doseFind

Inverse (dose-finding) estimate of a target x value (e.g., a percentile)

Description

Inverse ("dose-finding") point estimation of a dose (x) for a specified target y value (e.g., a response rate), using a user-specified forward-estimation algorithm (default is CIR).

Usage

```
doseFind(
  y,
  x = NULL,
  wt = NULL,
  estfun = cirPAVA,
  target = NULL,
  full = FALSE,
  dec = FALSE,
  extrapolate = FALSE,
  errOnFlat = FALSE,
  adaptiveShrink = FALSE,
  starget = target[1],
  ...
)
```

Arguments

y	can be either of the following: y values (response rates), a 2-column matrix with positive/negative response counts by dose, a DRtrace object or a doseResponse object.
x	dose levels (if not included in y).
wt	weights (if not included in y).

estfun	the name of the dose-response estimation function. Default cirPAVA .
target	A vector of target response rate(s), for which the percentile dose estimate is needed. See Note.
full	logical, is a more complete output desired (relevant only for doseFind)? if FALSE (default), only a point estimate of the dose (x) for the provided target rate is returned.
dec	(relevant only for doseFind) logical, is the true function is assumed to be monotone decreasing? Default FALSE.
extrapolate	logical: should extrapolation beyond the range of estimated y values be allowed? Default FALSE.
errOnFlat	logical: in case the forward estimate is completely flat making dose-finding infeasible, should an error be returned? Under default (FALSE), NAs are returned for the target estimate.
adaptiveShrink	logical, should the y-values be pre-shrunk towards an experiment's target? Recommended if data were obtained via an adaptive dose-finding design. See DRshrink and the Note.
starget	The shrinkage target. Defaults to target[1].
...	Other arguments passed on to doseResponse and estfun.

Details

The function works by calling estfun for forward estimation of the x-y relationship, then using [approx](#) with the x and y roles reversed for inverse estimation. The extrapolate option sets the rule argument for this second call:

- extrapolate=TRUE translates to rule=2, which actually means that the x value on the edge of the estimated y range will be assigned.
- extrapolate=FALSE (default) translates to rule=1, which means an NA will be returned for any target y value lying outside the estimated y range.

Note also that the function is set up to work with a vector of targets.

If the data were obtained from an adaptive dose-finding design and you seek to estimate a dose other than the experiment's target, note that away from the target the estimates are likely biased (Flournoy and Oron, 2019). Use adaptiveShrink=TRUE to mitigate the bias. In addition, either provide the true target as starget, or a vector of values to target, with the first value being the true target.

Value

under default, returns point estimate(s) of the dose (x) for the provided target rate(s). With full=TRUE, returns a list with

- targest The said point estimate of x
- input a doseResponse object summarizing the input data
- output a doseResponse object with the forward estimate at design points
- shrinkage a doseResponse object which is the alg output of the forward-estimation function

Author(s)

Assaf P. Oron <assaf.aron.at.gmail.com>

References

Flournoy N and Oron AP, 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

See Also

[oldPAVA](#), [cirPAVA](#). If you'd like point and interval estimates together, use [quickInverse](#).

DRshrink

Shrinkage fix to bias in observed rates under adaptive dose-finding design

Description

Adaptive dose-finding designs induce a bias on observed rates, away from the target dose. This is well-known in other adaptive-design fields, but has been overlooked by the dose-finding research community. Flournoy and Oron (2020) examine the bias in the dose-finding context, and suggest a simple shrinkage fix that reduces both bias and variance. The fix is analogous to the empirical-logit fix to binary data, but instead of adding 0.5 to each cell, `target` is added to the 1's at each dose, and `1-target` to the 0's. The shrinkage is applied to the raw observation, so CIR or IR are carried out on the shrunk data.

Usage

```
DRshrink(y, x = NULL, wt0 = NULL, target, swt = 1, nmin = 2, ...)
```

Arguments

<code>y</code>	can be either of the following: <code>y</code> values (response rates), a 2-column matrix with positive/negative response counts by dose, a DRtrace object or a doseResponse object.
<code>x</code>	dose levels (if not included in <code>y</code>).
<code>wt0</code>	weights (if not included in <code>y</code>).
<code>target</code>	the balance point (between 0 and 1) around which the design concentrates allocations.
<code>swt</code>	the weight of the shrinkage. Default 1 (a single observation)
<code>nmin</code>	the minimum <code>n</code> at each dose, for the shrinkage to be applied. Default 2.
<code>...</code>	parameters passed on to <code>doseResponse()</code>

Author(s)

Assaf P. Oron <assaf.aron.at.gmail.com>

References

Flournoy N and Oron AP, 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

is.DRtrace	<i>Constructor functions and class-checking functions for DRtrace and doseResponse classes</i>
------------	--

Description

Functions to create and sanity-check objects of the `DRtrace` (dose-response experiment trace/trajectory) and `doseResponse` (dose-response raw summary) classes. Note that the latter inherits from the former, purely for programming-convenience reasons.

Usage

```
is.DRtrace(dr)
```

```
is.doseResponse(dr)
```

```
DRtrace(y, x = NULL, cohort = NULL, noyes = FALSE, ...)
```

```
doseResponse(y, x = NULL, wt = rep(1, length(y)), noyes = FALSE, ...)
```

Arguments

<code>dr</code>	the object being checked
<code>y, x</code>	see Details.
<code>cohort</code>	(<code>DRtrace</code> only) specify each observation's cohorts, if there were cohorts. If all cohorts were the same size, then you can specify the size as a single number. If there were no cohorts, code will default this variable to <code>1:n</code>
<code>noyes</code>	logical, in case of a 2-column input is the 1st column 'no'? Default <code>FALSE</code> , meaning the 1st column is 'yes'.
<code>...</code>	parameters passed on to <code>DRtrace()</code> , or ignored.
<code>wt</code>	(<code>doseResponse</code> only) the weights associated with each <code>x</code> value; usually the sample size or similar.

Details

The input argument `y` can include the entire information, or as little as the `y` vector of responses (for a `DRtrace` object) or response rates (`doseResponse`). When including the entire information, it has to be a data frame with at least `y` (both `y` and `x` for `DRtrace`), or a two-column matrix with 'yes' and 'no' responses (assumed in this order, but can be the reverse with `noyes=TRUE`). In this case the doses `x` can be provided as a separate vector, or as the matrix row names. `doseResponse` will return an error if there are any duplicates in `x`.

Even though both `DRtrace` and `doseResponse` accept two-column yes/no matrix input, the interpretation is different. For the former, this form of input is intended mostly to enable shorthand input when the experiment was run in cohorts. Each row represents a cohort's results, and rows must be in the order the experiment was run. For the latter, the yes-no table is a summary tabulation of responses and is treated accordingly, including rearrangement of rows to increasing `x`.

Value

For constructor functions, the relevant object. For checking functions, a logical value indicating whether the object meets class definition.

Author(s)

Assaf P. Oron <assaf.oron.at.gmail.com>

See Also

[cirPAVA](#), [plot.doseResponse](#), [plot.DRtrace](#)

Examples

```
## Summary of raw data from the notorious Neuenschwander et al. (Stat. Med., 2008) trial
## Note the use of the 'cohort' argument to specify the cohort order
neundatTrace = DRtrace(x = c(rep(1:4,each=4), 7, 7, rep(6,9)),
                      y = c(rep(0,16), 1,1, rep(c(0,0,1),2), 0,0,0),
                      cohort = rep(1:8, c(4,4,4,4, 2, 3,3,3)) )
par(mar=c(3,3,3,1), mgp=c(2,.5,0), tcl=-0.25)
layout(t(1:2))
plot(neundatTrace ,main="N. et al. (2008) Cohort Trace", xlab = 'Cohort',
     ylab="Ordinal Dose Level" ,cex.main=1.5)

## Same data, in 'doseResponse' format with actual doses rather than dose levels
neundatDose = doseResponse(x=c(1,2.5,5,10,20,25), y = c(rep(0,4),2/9,1), wt = c(3,4,5,4,9,2) )
plot(neundatDose ,main="N. et al. (2008) Final Dose-Toxicity", ylim=c(0,1),
     xlab="Dose (mg/sq.m./wk)", ylab="Toxicity Response Curve (F)", cex.main=1.5)
## We can also convert the DRtrace object to doseResponse...
neundatLevel = doseResponse(neundatTrace)

### Now plotting the former, vs. IR/CIR estimates
neunCIR0 = cirPAVA(neundatDose,full=TRUE, adaptiveShrink = TRUE, target = 0.3)
lines(neunCIR0$shrinkage$x, neunCIR0$shrinkage$y, type='b' ,pch=19)
legend(1,1, pch=c(4,19), legend=c('Observations', 'CIR w/bias corr.'), bty='n')
```

isotInterval

Returns analytical interval estimates, given isotonic-regression (centered or not) point estimates

Description

For confidence intervals at design points (x values with observations), this function calls `intfun` to do the work. In addition, CIs for any x value are calculated using linear interpolation between design points (note that for CIR, this differs from the interpolation of point estimates which is carried out between shrinkage points, as explained in [quickIsotone](#))

Usage

```
isotInterval(
  isotPoint,
  outx = isotPoint$output$x,
  conf = 0.9,
  intfun = morrisCI,
  ...
)
```

Arguments

<code>isotPoint</code>	The output of an estimation function such as <code>cirPAVA</code> with the option <code>full=TRUE</code> . Should be a list of 3 <code>doseResponse</code> objects named <code>input</code> , <code>output</code> , <code>shrinkage</code> .
<code>outx</code>	vector of x values for which estimates will be made. If <code>NULL</code> (default), this will be set to the set of unique values in <code>isotPoint\$x</code> argument (or equivalently in <code>y</code>).
<code>conf</code>	numeric, the interval's confidence level as a fraction in $(0,1)$. Default 0.9.
<code>intfun</code>	the function to be used for interval estimation. Default <code>morrisCI</code> (see help on that function for additional options).
<code>...</code>	additional arguments passed on to <code>intfun</code>

Value

a data frame with two variables `ciLow`, `ciHigh` containing the estimated lower and upper confidence bounds, respectively.

Note

All provided algorithm and formulae are for Binomial data only. For other data, write your own `intfun`, returning a two-column matrix. The interval estimation method is presented and discussed by Oron and Flourmoy (2017).

Interval coverage for extreme percentiles with adaptive designs may be lacking: use `adaptiveCurve=TRUE` whenever the target is not 0.5. However, targeting the 5th or 95th percentile will likely produce intervals with 10-15% under-coverage by with that option.

Author(s)

Assaf P. Oron <assaf.aron.at.gmail.com>

References

Oron, A.P. and Flournoy, N., 2017. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. *Statistics in Biopharmaceutical Research* 3, 258-267.

See Also

[quickIsotone](#), [quickInverse](#), [morrisCI](#),

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# The experiment's goal is to find the 30th percentile
slow1=cirPAVA(dat,full=TRUE)
# Default interval (Morris+Wilson); same as you get by directly calling 'quickIsotone'
int1=isotInterval(slow1)
# Morris without Wilson; the 'narrower=FALSE' argument is passed on to 'morrisCI'
int1_0=isotInterval(slow1,narrower=FALSE)
# Wilson without Morris
int2=isotInterval(slow1,intfun=wilsonCI)
# Agresti-Coull (the often-used "plus 2")
int3=isotInterval(slow1,intfun=agcouCI)
# Jeffrys (Bayesian-inspired) is also available
int4=isotInterval(slow1,intfun=jeffCI)

### Showing the data and the intervals
par(mar=c(3,3,4,1),mgp=c(2,.5,0),tcl=-0.25)
plot(dat,ylim=c(0,0.65),refsize=4,las=1,main="Forward-Estimation CIs") # uses plot.doseResponse()

# The true response function; true target is where it crosses the y=0.3 line
lines(seq(0,7,0.1),pweibull(seq(0,7,0.1),shape=1.1615,scale=8.4839),col=4)

lines(int1$ciLow,lty=2,col=2,lwd=2)
lines(int1$ciHigh,lty=2,col=2,lwd=2)

lines(int1_0$ciLow,lty=2)
lines(int1_0$ciHigh,lty=2)

lines(int2$ciLow,lty=2,col=3)
lines(int2$ciHigh,lty=2,col=3)
# Plotting the remaining 2 is skipped, as they are very similar to Wilson.

# Note how the default (red) boundaries take the tighter of the two options everywhere,
# except for one place (dose 1 upper bound) where they go even tighter thanks to monotonicity
# enforcement. This can often happen when sample size is uneven; since bounds tend to be
# conservative it is rather safe to do.

legend('topleft',pch=c(NA,'X',NA,NA,NA),lty=c(1,NA,2,2,2),col=c(4,1,2,1,3),lwd=c(1,1,2,1,1),legend
=c('True Curve','Observations','Morris+Wilson (default)','Morris only','Wilson only'),bty='n')
```

iterCIR	<i>Centered-isotonic-regression (CIR): iterative version for better bias correction</i>
---------	---

Description

EXPERIMENTAL: Nonparametric forward point estimation of a monotone response (y) as a function of dose (x), using an iterative version of the centered-isotonic-regression (CIR) algorithm. The code works, but delivers marginal improvement at greater computational cost (an issue if you simulate a large ensemble), and somewhat convoluted interpretation. Use at your own risk. For explanation, see Oron and Flournoy (2017), Section 3.2.

Usage

```
iterCIR(y, outx = NULL, tol = 0.001, maxit = 10, full = FALSE, ...)
```

Arguments

<code>y</code>	See cirPAVA
<code>outx</code>	vector of x values for which predictions will be made. If <code>NULL</code> (default), this will be set to the set of unique values in the x argument (or equivalently in $y\$x$).
<code>tol</code>	The iteration's convergence tolerance level (default 1e-3)
<code>maxit</code>	integer, maximum number of iterations (default 10)
<code>full</code>	logical, is a more complete output desired? if <code>FALSE</code> (default), only a vector of point estimates for y at the provided dose levels is returned
<code>...</code>	Other arguments passed on to cirPAVA

Value

under default, returns a vector of y estimates at unique x values. With `full=TRUE`, returns a list of 3 [doseResponse](#) objects named `output`, `input`, `shrinkage` for the output data at dose levels, the input data, and the function as fit at algorithm-generated points, respectively.

See Also

[cirPAVA](#), [quickIsotone](#)

 morrisCI

Analytical confidence intervals using the Morris (1988) algorithm

Description

Analytical confidence intervals for CIR and IR, using the recursive algorithm by Morris (1988), equation (4.3), for ordered-binomial point estimates. Optionally, the intervals are narrowed further using a backup pointwise interval estimate.

Usage

```

morrisCI(
  y,
  n,
  phat = y/n,
  conf = 0.9,
  narrower = TRUE,
  alternate = wilsonCI,
  ...
)

```

Arguments

y	integer or numeric vector, the pointwise Binomial counts
n	integer or numeric vector, the pointwise sample sizes
phat	numeric vector, the point estimates. Defaults to y/n , but when called by isotInterval is overridden by the actual CIR/IR point estimate.
conf	numeric, the interval's confidence level as a fraction in (0,1). Default 0.9.
narrower	logical, if the alternate-produced interval is narrower at any point, should it replace the Morris result? Also, can we enforce straightforward monotonicity to narrow the bounds? Default TRUE.
alternate	function to use for alternate pointwise interval. Default wilconCI.
...	parameters passed on to alternate.

Details

The default for backup is Wilson's ([wilconCI](#)). Also available are Jeffrys' ([jeffCI](#)) and Agresti-Coull ([agcouCI](#)).

Value

A two-column matrix with the same number of rows as `length(phat)`, containing the calculated lower and upper bounds, respectively.

Note

This function found and corrected a typo in equation (4.3), namely the use of $G_{(j+1)}$ in the recursion. The recursion cannot start in this way. Rather, it is the use of $\theta_{(j+1)}$ that delivers information from adjacent doses. Or perhaps in other words, there is only one G function rather than a different one for each dose. The correction has been verified by reproducing the numbers in the Morris (1988) example (Table 1), and also approved by the original author.

Author(s)

Assaf P. Oron <assaf.oron.at.gmail.com>

References

Morris, M., 1988. Small-sample confidence limits for parameters under inequality constraints with application to quantal bioassay. *Biometrics* 44, 1083-1092.

See Also

[isotInterval](#)

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# The experiment's goal is to find the 30th percentile
slow1=cirPAVA(dat,full=TRUE)
# Default interval (Morris+Wilson); same as you get by directly calling 'quickIsotone'
int1=isotInterval(slow1)
# Morris without Wilson; the 'narrower=FALSE' argument is passed on to 'morrisCI'
int1_0=isotInterval(slow1,narrower=FALSE)
# Wilson without Morris
int2=isotInterval(slow1,intfun=wilsonCI)
# Agresti=Coull (the often-used "plus 2")
int3=isotInterval(slow1,intfun=agcouCI)
# Jeffrys (Bayesian-inspired) is also available
int4=isotInterval(slow1,intfun=jeffCI)

### Showing the data and the intervals
par(mar=c(3,3,4,1),mgp=c(2,.5,0),tcl=-0.25)
plot(dat,ylim=c(0,0.65),refsize=4,las=1,main="Forward-Estimation CIs") # uses plot.doseResponse()

# The true response function; true target is where it crosses the y=0.3 line
lines(seq(0,7,0.1),pweibull(seq(0,7,0.1),shape=1.1615,scale=8.4839),col=4)

lines(int1$ciLow,lty=2,col=2,lwd=2)
lines(int1$ciHigh,lty=2,col=2,lwd=2)

lines(int1_0$ciLow,lty=2)
lines(int1_0$ciHigh,lty=2)
```

```

lines(int2$ciLow,lty=2,col=3)
lines(int2$ciHigh,lty=2,col=3)
# Plotting the remaining 2 is skipped, as they are very similar to Wilson.

# Note how the default (red) boundaries take the tighter of the two options everywhere,
# except for one place (dose 1 upper bound) where they go even tighter thanks to monotonicity
# enforcement. This can often happen when sample size is uneven; since bounds tend to be
# conservative it is rather safe to do.

legend('topleft',pch=c(NA,'X',NA,NA,NA),lty=c(1,NA,2,2,2),col=c(4,1,2,1,3),lwd=c(1,1,2,1,1),legend
=c('True Curve','Observations','Morris+Wilson (default)','Morris only','Wilson only'),bty='n')

```

oldPAVA	<i>Returns standard isotonic-regression estimate, with flexible dose-response input</i>
---------	---

Description

Nonparametric forward point estimation of a monotone response (y), using the standard isotonic-regression pool-adjacent-violators algorithm (PAVA). Core code from Raubertas (1994) with many modifications.

Usage

```

oldPAVA(
  y,
  x = NULL,
  wt = rep(1, length(x)),
  outx = NULL,
  full = FALSE,
  dec = FALSE,
  adaptiveShrink = FALSE,
  ...
)

```

Arguments

y	can be either of the following: y values (response rates), a 2-column matrix with positive/negative response counts by dose, a DRtrace object or a doseResponse object.
x	dose levels (if not included in y). Note that the PAV algorithm doesn't really use them.
wt	weights (if not included in y).
$outx$	vector of x values for which predictions will be made. If NULL (default), this will be set to the set of unique values in the x argument (or equivalently in $y\$x$). Non-NULL inputs are relevant only if <code>full=TRUE</code> .

full	logical, is a more complete output desired? if FALSE (default), only a vector of point estimates for y at the provided dose levels is returned
dec	logical, is the true function is assumed to be monotone decreasing? Default FALSE.
adaptiveShrink	logical, should the y-values be pre-shrunk towards an experimental target? May be relevant if data were obtain via an adaptive dose-finding design. See DRshrink .
...	Other arguments passed on to the constructor functions that pre-process the input.

Details

Compute the isotonic regression of a numeric vector 'y', with weights 'wt', with respect to simple order. The core algorithm is still the one coded by R.F. Raubertas, dated 02 Sep 1994. However, the input and output modules have been modified to allow more flexible formats in either direction. note that unlike centered-isotonic-regression (CIR, see [cirPAVA](#)), this algorithm does not use the dose (x) values at all. For a discussion why CIR is preferred over "plain-vanilla" PAVA, see Oron and Flournoy (2017).

Value

under default, returns a vector of y estimates at unique x values. With full=TRUE, returns a list of 3 [doseResponse](#) objects named output, input, shrinkage for the output data at dose levels, the input data, and the function as fit at algorithm-generated points, respectively. For this function, the first and third objects are identical.

Author(s)

C.R. Raubertas, Assaf P. Oron <assaf.oron.at.gmail.com>

References

Oron, A.P. and Flournoy, N., 2017. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. Statistics in Biopharmaceutical Research, In Press (author's public version available on [arxiv.org](#)).

See Also

[cirPAVA](#)

plot.DRtrace

Plotting Methods for DRtrace, doseResponse Objects

Description

Plotting methods for [doseResponse](#) and [DRtrace](#) classes.

Usage

```
## S3 method for class 'DRtrace'
plot(
  x,
  xlab = "Patient Order",
  ylab = "Dose",
  shape = "circle",
  connect = TRUE,
  mcol = 1,
  dosevals = NULL,
  offset = 0.2,
  ...
)

## S3 method for class 'doseResponse'
plot(
  x,
  xlab = "Dose",
  ylab = "Response",
  pch = "X",
  varsize = TRUE,
  refsize = sqrt(1/mean(x$weight)),
  connect = FALSE,
  mcol = 1,
  dosevals = NULL,
  ...
)
```

Arguments

x	the object, whether DRtrace or doseResponse
xlab, ylab	x-axis and y-axis labels passed on to plot
shape	the plotting shape (DRtrace only): 'circle' (default), 'square', or 'triangle'
connect	logical: whether to connect the symbols (generic plotting type 'b'). Default TRUE for DRtrace and FALSE for doseResponse
mcol	The color of the main plotting symbols and connecting lines. Default 1 (the current palette's first color). Note: if you change the color and inadvertently use col instead, there will be an error message.
dosevals	Dose values to be plotted along the x-axis (plot.doseResponse) or y-axis (plot.DRtrace) . If NULL (default), those will be the doses in the dataset (i.e.,sort(unique(x\$x))).
offset	(DRtrace only) In case of a cohort-based experiment, the relative vertical offset between symbols for outcomes within the same cohort (as fraction of dose spacing). Default 0.2.
...	Other arguments passed on to plot .

	Conversely, putting values on a different scale into <code>dosevals</code> , or even text labels instead of numbers, won't work. For the former, change the scale at the source data (i.e., in the plotted object). For the latter, sorry but no solution at present.
<code>pch</code>	the plotting character (<code>doseResponse</code> only), the default being 'X' marks
<code>varsize</code>	(<code>doseResponse</code> only) logical, should symbol size vary by sample size? Default TRUE
<code>refsize</code>	(<code>doseResponse</code> only) a reference size by which the plotting sizes will be multiplied. Default is $1/\sqrt{\text{mean}(\text{dr}\$weight)}$, scaled so that if <code>varsize = TRUE</code> the weighted-average symbol size is 1. If <code>varsize = FALSE</code> , this argument is equivalent to <code>cex</code> in an ordinary x-y <code>plot()</code> call.

Details

Generic methods for dose-response trajectory/trace (`DRtrace`), and dose-response summary (`doseResponse`) class objects. The `DRtrace` plotting uses the typical convention of plotting dose-finding experimental trace, with dose levels (x) in the vertical axis and 1/0 responses (y) denoted via filled/empty circles, respectively. In other words, this generic plotting method is only relevant for binary 0/1 outcomes. The `doseResponse` plotting has response rate on the y-axis and dose on the x-axis, and plots symbols whose area is proportional to the weights.

Author(s)

Assaf P. Oron <assaf.oron.at.gmail.com>

See Also

[doseResponse](#), [DRtrace](#)

Examples

```
## Summary of raw data from the notorious Neuenschwander et al. (Stat. Med., 2008) trial
## Note the use of the 'cohort' argument to specify the cohort order
neundatTrace = DRtrace(x = c(rep(1:4,each=4), 7, 7, rep(6,9)),
                      y = c(rep(0,16), 1,1, rep(c(0,0,1),2), 0,0,0),
                      cohort = rep(1:8, c(4,4,4,4, 2, 3,3,3)) )
par(mar=c(3,3,3,1), mgp=c(2,.5,0), tcl=-0.25)
layout(t(1:2))
plot(neundatTrace ,main="N. et al. (2008) Cohort Trace", xlab = 'Cohort',
      ylab="Ordinal Dose Level" ,cex.main=1.5)

## Same data, in 'doseResponse' format with actual doses rather than dose levels
neundatDose = doseResponse(x=c(1,2.5,5,10,20,25), y = c(rep(0,4),2/9,1), wt = c(3,4,5,4,9,2) )
plot(neundatDose ,main="N. et al. (2008) Final Dose-Toxicity", ylim=c(0,1),
      xlab="Dose (mg/sq.m./wk)", ylab="Toxicity Response Curve (F)", cex.main=1.5)
## We can also convert the DRtrace object to doseResponse...
neundatLevel = doseResponse(neundatTrace)

### Now plotting the former, vs. IR/CIR estimates
neunCIR0 = cirPAVA(neundatDose,full=TRUE, adaptiveShrink = TRUE, target = 0.3)
lines(neunCIR0$shrinkage$x, neunCIR0$shrinkage$y, type='b' ,pch=19)
legend(1,1, pch=c(4,19), legend=c('Observations', 'CIR w/bias corr.'), bty='n')
```

quickInverse	<i>Point and Interval Inverse Estimation ("Dose-Finding"), using CIR and IR</i>
--------------	---

Description

Convenience wrapper for point and interval estimation of the "dose" that would generate a target "response" value, using CIR and IR.

Usage

```
quickInverse(
  y,
  x = NULL,
  wt = NULL,
  target,
  estfun = cirPAVA,
  intfun = morrisCI,
  delta = TRUE,
  conf = 0.9,
  resolution = 100,
  extrapolate = FALSE,
  adaptiveShrink = FALSE,
  starget = target[1],
  adaptiveCurve = FALSE,
  ...
)
```

Arguments

y	can be either of the following: y values (response rates), a 2-column matrix with positive/negative response counts by dose, a DRtrace object or a doseResponse object.
x	dose levels (if not included in y).
wt	weights (if not included in y).
target	A vector of target response rate(s), for which the percentile dose estimate is needed. See Note.
estfun	the function to be used for point estimation. Default cirPAVA .
intfun	the function to be used for interval estimation. Default morrisCI (see help on that function for additional options).
delta	logical: should intervals be calculated using the delta ("local") method (default, TRUE) or back-drawn directly from the forward bounds? See Details.
conf	numeric, the interval's confidence level as a fraction in (0,1). Default 0.9.
resolution	numeric: how fine should the grid for the inverse-interval approximation be? Default 100, which seems to be quite enough. See 'Details'.

extrapolate	logical: should extrapolation beyond the range of estimated y values be allowed? Default FALSE. Note this affects only the point estimate; interval boundaries are not extrapolated.
adaptiveShrink	logical, should the y-values be pre-shrunk towards an experiment's target? Recommended when the data were obtained via an adaptive dose-finding design. See DRshrink and the Note below.
starget	The shrinkage target. Defaults to target[1].
adaptiveCurve	logical, should the CIs be expanded by using a parabolic curve between estimation points rather than straight interpolation (default FALSE)? Recommended when adaptive design was used and target is not 0.5.
...	Other arguments passed on to doseFind and quickIsotone , and onwards from there.

Details

The inverse point estimate is calculated in a straightforward manner from a forward estimate, using [doseFind](#). For the inverse interval, the default option (`delta=TRUE`) calls [deltaInverse](#) for a "local" (Delta) inversion of the forward intervals. If `delta=FALSE`, a second call to [quickIsotone](#) generates a high-resolution grid outlining the forward intervals. Then the algorithm "draws a horizontal line" at `y=target` to find the right and left bounds on this grid. Note that the right (upper) dose-finding confidence bound is found on the lower forward confidence bound, and vice versa. This approach is not recommended, tending to produce CIs that are too wide.

If the data were obtained from an adaptive dose-finding design and you seek to estimate a dose other than the experiment's target, note that away from the target the estimates are likely biased (Flournoy and Oron, 2019). Use `adaptiveShrink=TRUE` to mitigate the bias. In addition, either provide the true target as `starget`, or a vector of values to `target`, with the first value being the true target.

Value

A data frame with 4 elements:

- `target` The user-provided target values of y, at which x is estimated
- `point` The point estimates of x
- `lowerPPconf`, `upperPPconf` the interval-boundary estimates for a 'PP'=100*conf confidence interval

References

Flournoy N and Oron AP, 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

See Also

[quickIsotone](#), [doseFind](#), [deltaInverse](#)

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# The experiment's goal is to find the 30th percentile
inv1=quickInverse(dat, target=0.3, adaptiveShrink = TRUE, adaptiveCurve = TRUE)
# With old PAVA as the forward estimator, and without the adaptive-design corrections:
inv0=quickInverse(dat, target=0.3, estfun=oldPAVA)

### Showing the data and the estimates
par(mar=c(3,3,1,1), mgp=c(2,.5,0), tcl=-0.25)
plot(dat, ylim=c(0.05,0.55), las=1) # uses plot.doseResponse()

# The true response function; true target is where it crosses the y=0.3 line
lines(seq(1,5,0.1),pweibull(seq(1,5,0.1),shape=1.1615,scale=8.4839),col=4)
abline(h=0.3,col=2,lty=3)
# Plotting the point estimates, as "tick" marks on the y=0.3 line
lines(rep(inv1$point,2),c(0.25,0.35), lwd=1.5) # CIR
lines(rep(inv0$point,2),c(0.25,0.35),lty=2, lwd=1.5) # IR
# You could plot the CIs too,
# Here's code to plot the CIR 90% CI as a light-green rectangle:
# rect(inv1$lower90conf,0.25,inv1$upper90conf,0.35,col=rgb(0,1,0,alpha=0.3),border=NA)
# Intervals are plotted and interval options are explored more extensively
#      in the 'deltaInverse' help page.

legend('topleft',pch=c(NA,'X',NA,NA),lty=c(1,NA,2,1),col=c(4,1,1,1),
legend=c('True Curve','Observations','IR Estimate','CIR Estimate'),bty='n')
```

quickIsotone

One-Stop-shop Forward point and interval estimation via CIR or IR

Description

One-Stop-shop Forward point and confidence-interval estimation of a monotone response (y) as a function of dose (x), using centered-isotonic-regression (CIR, default) or isotonic regression. Input format is rather flexible. This function calls `cirPAVA`, `oldPAVA`, `iterCIR` (speculatively), or a user-written function, for the point estimate, then `isotInterval` for the confidence interval. Vector input is allowed, but the preferred input format is a `doseResponse` object. An analogous function for dose-finding (inverse estimation) is `quickInverse`.

Usage

```
quickIsotone(
  y,
  x = NULL,
  wt = NULL,
  outx = NULL,
```



```

    dec = FALSE,
    estfun = cirPAVA,
    intfun = morrisCI,
    conf = 0.9,
    adaptiveShrink = FALSE,
    ...
)

```

Arguments

y	can be either of the following: y values (response rates), a 2-column matrix with positive/negative response counts by dose, a DRtrace object or a doseResponse object.
x	dose levels (if not included in y). Note that the PAV algorithm doesn't really use them.
wt	weights (if not included in y).
outx	vector of x values for which predictions will be made. If NULL (default), this will be set to the set of unique values in the x argument (or equivalently in y\$x).
dec	logical, is the true function assumed to be monotone decreasing rather than increasing? Default FALSE.
estfun	the function to be used for point estimation. Default cirPAVA .
intfun	the function to be used for interval estimation. Default wilsonCI (see help on that function for additional options).
conf	numeric, the interval's confidence level as a fraction in (0,1). Default 0.9.
adaptiveShrink	logical, should the y-values be pre-shrunk towards an experiment's target? Recommended if data were obtained via an adaptive dose-finding design. If TRUE, then must also provide a target argument that will be passed via ...
...	arguments passed on to other functions (constructor, point estimate and interval estimate).

Value

A data frame with 4 variables:

- x either the input x values, or outx of specified;
- y The point estimates of x
- lowerPPconf , upperPPconf the interval-boundary estimates for a 'PP'=100*conf confidence interval

Note

You can obtain interpolated point estimates for x values between the observed data by specifying them via outx. However, for CIR, do NOT commit the error of generating estimates at observations, then interpolating using [approx](#). If you need to retain a set of estimates for plotting the entire fitted curve, or for future interpolation at unknown points, call [cirPAVA](#) directly with full=TRUE, then use the returned shrinkage data frame for plotting and interpolation. See example code below.

If the data were obtained from an adaptive dose-finding design then away from the design's target the estimates are likely biased (Flournoy and Oron, 2020). Use `adaptiveShrink=TRUE` to mitigate the bias.

Author(s)

Assaf P. Oron <assaf.oron.at.gmail.com>

References

Oron, A.P. and Flournoy, N., 2017. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. *Statistics in Biopharmaceutical Research* 9, 258-267. (author's public version available on arxiv.org).

Flournoy, N. and Oron, A.P., 2020. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 47, 2431-2442.

See Also

[cirPAVA](#), [oldPAVA](#), [isotInterval](#), [quickInverse](#), [doseResponse](#)

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# CIR, using the default 'quick' function that also provides CIs (default 90%).
# The experiment's goal is to find the 30th percentile. We deploy the empirical bias correction.
quick1=quickIsotone(dat, adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
quick1
# Use 'estfun' argument to operate the same function with old PAVA as the estimator
# Here we neglect the bias correction to sharpen the old:new contrast
quick0=quickIsotone(dat,estfun=oldPAVA)
quick0

### Showing the data and the fits
par(mar=c(3,3,1,1),mgp=c(2,.5,0),tcl=-0.25)
plot(dat, ylim=c(0.05,0.55), las=1) # uses plot.doseResponse()
# The IR fit: a straightforward interpolation
lines(quick0$y,lty=2)

# With CIR, 'quickIsotone' cannot show us the true underlying interpolation;
# it only provides the estimates at requested points. Interpolation should be done between
# shrinkage points, not the original design points. So we must call the full 'cirPAVA' function:

slow1 = cirPAVA(dat, full=TRUE, adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
# Now, compare these 3 (the first one is wrong, b/c it interpolates from design points):
midpts = 1:4 + 0.5
approx(1:5,quick1$y, xout=midpts)$y
# instead, you can just call 'quickIsotone' and specify 'outx'
quickIsotone(dat,outx=midpts , adaptiveShrink = TRUE, adaptiveCurve = TRUE, target = 0.3)
approx(slow1$shrinkage$x,slow1$shrinkage$y,xout=midpts)$y # Or use 'cirPAVA'
```

```
# Ok... finally plotting the CIR curve
# Both flat intervals are shrunk, because neither are at y=0 or y=1
lines(slow1$shrinkage$x,slow1$shrinkage$y, lwd = 2)

# Last but not least, here's the true response function
lines(seq(1,5,0.1),pweibull(seq(1,5,0.1),shape=1.1615,scale=8.4839),col=2)
legend('topleft',pch=c(NA,'X',NA,NA),lty=c(1,NA,2,1),col=c(2,1,1,1),
legend=c('True Curve','Observations','IR','CIR'), bty='n')
```

slope	<i>Piecewise-linear local slopes given a (non-strictly) monotone x-y sequence</i>
-------	---

Description

Estimate monotone piecewise-linear slopes, with the default behavior forbidding zero slope. This behavior is due to the fact that the function is used to invert confidence intervals using the Delta method. The input interval has to be strictly increasing in x, and (non-strictly) monotone in y (increasing or decreasing).

Usage

```
slope(
  x,
  y,
  outx = x,
  allowZero = FALSE,
  tol = 0.01,
  full = FALSE,
  decreasing = FALSE
)
```

Arguments

x	numeric or integer: input x values, must be strictly increasing
y	numeric: input y values, must be monotone (can be non-strict) and in line with the direction specified by decreasing
outx	numeric or integer: x values at which slopes are desired (default: same as input values)
allowZero	logical: should zero be allowed in the output? Default FALSE
tol	tolerance level: when allowZero=FALSE, slope below that value is considered zero. Default 1e-2. Might need to change if you use unusual units for x or y.
full	logical: should a more detailed output be provided? Default FALSE (see details under 'Value').
decreasing	logical: is input supposed to be monotone decreasing rather than increasing? Default FALSE

Details

At design points (i.e., the input x values), the function takes the average between the left and right slopes (on the edges the inside slope is technically replicated to the outside). If `allowZero=FALSE` (default), the algorithm gradually expands the x range over which slope is observed (by increments of one average x spacing), until a positive slope results. If the input is completely flat in y and `allowZero=FALSE`, the function returns NAs.

Value

If `full=FALSE`, returns a vector of slopes at the points specified by `outx`.

If `full=TRUE`, returns a list with slopes at the design point (`rawslopes`), the initial guess at output slopes (`initial`), and the official final ones (`final`).

See Also

[deltaInverse](#), which uses this function.

wilsonCI

Standard unordered-Binomial confidence interval utilities.

Description

Standard small-sample Binomial confidence interval utilities, using the methods of Wilson, Agresti-Coull and Jeffrys.

Usage

```
wilsonCI(phat, n, conf = 0.9, ...)
```

```
agcouCI(phat, n, conf = 0.9, ...)
```

```
jeffCI(phat, n, conf = 0.9, w1 = 0.5, w2 = w1, ...)
```

Arguments

<code>phat</code>	numeric vector, point estimates for which an interval is sought
<code>n</code>	integer vector of same length, of pointwise sample sizes
<code>conf</code>	numeric in (0,1), the confidence level
<code>...</code>	pass-through for compatibility with a variety of calling functions
<code>w1, w2</code>	numeric, weights used in <code>jeffCI</code> only

Details

These functions implement the basic (uncorrected) three intervals which are seen by the consensus of literature as the "safest" off-the-shelf formulae. None of them account for ordering or monotonicity; therefore the `cir` package default is `morrisCI` which does account for that, with the 3 unordered formulae used for optional narrowing of the interval at individual points.

Value

A two-column matrix with the same number of rows as `length(phat)`, containing the calculated lower and upper bounds, respectively.

See Also

[isotInterval](#) for more details about how forward CIs are calculated, [quickInverse](#) for inverse (dose-finding) intervals.

Examples

```
# Interesting run (#664) from a simulated up-and-down ensemble:
# (x will be auto-generated as dose levels 1:5)
dat=doseResponse(y=c(1/7,1/8,1/2,1/4,4/17),wt=c(7,24,20,12,17))
# The experiment's goal is to find the 30th percentile
slow1=cirPAVA(dat,full=TRUE)
# Default interval (Morris+Wilson); same as you get by directly calling 'quickIsotone'
int1=isotInterval(slow1)
# Morris without Wilson; the 'narrower=FALSE' argument is passed on to 'morrisCI'
int1_0=isotInterval(slow1,narrower=FALSE)
# Wilson without Morris
int2=isotInterval(slow1,intfun=wilsonCI)
# Agresti-Coull (the often-used "plus 2")
int3=isotInterval(slow1,intfun=agcouCI)
# Jeffrys (Bayesian-inspired) is also available
int4=isotInterval(slow1,intfun=jeffCI)

### Showing the data and the intervals
par(mar=c(3,3,4,1),mgp=c(2,.5,0),tcl=-0.25)
plot(dat,ylim=c(0,0.65),refsize=4,las=1,main="Forward-Estimation CIs") # uses plot.doseResponse()

# The true response function; true target is where it crosses the y=0.3 line
lines(seq(0,7,0.1),pweibull(seq(0,7,0.1),shape=1.1615,scale=8.4839),col=4)

lines(int1$ciLow,lty=2,col=2,lwd=2)
lines(int1$ciHigh,lty=2,col=2,lwd=2)

lines(int1_0$ciLow,lty=2)
lines(int1_0$ciHigh,lty=2)

lines(int2$ciLow,lty=2,col=3)
lines(int2$ciHigh,lty=2,col=3)
# Plotting the remaining 2 is skipped, as they are very similar to Wilson.

# Note how the default (red) boundaries take the tighter of the two options everywhere,
# except for one place (dose 1 upper bound) where they go even tighter thanks to monotonicity
# enforcement. This can often happen when sample size is uneven; since bounds tend to be
# conservative it is rather safe to do.

legend('topleft',pch=c(NA,'X',NA,NA,NA),lty=c(1,NA,2,2,2),col=c(4,1,2,1,3),lwd=c(1,1,2,1,1),legend
=c('True Curve','Observations','Morris+Wilson (default)','Morris only','Wilson only'),bty='n')
```

Index

agcouCI (wilsonCI), 28
approx, 9, 25

cir (cir-package), 2
cir-package, 2
cirPAVA, 3, 6, 9, 10, 12, 13, 15, 19, 22, 24–26

deltaInverse, 5, 23, 28
doseFind, 6, 8, 23
doseResponse, 2–4, 6, 8–10, 13, 15, 18–22, 24–26
doseResponse (is.DRtrace), 11
DRshrink, 5, 7, 9, 10, 19, 23
DRtrace, 2, 3, 8, 10, 18–22, 25
DRtrace (is.DRtrace), 11

is.doseResponse (is.DRtrace), 11
is.DRtrace, 11
isotInterval, 7, 12, 16, 17, 24, 26, 29
iterCIR, 15, 24

jeffCI (wilsonCI), 28

morrisCI, 6, 13, 14, 16, 22, 28

oldPAVA, 5, 10, 18, 24, 26

plot, 20
plot.doseResponse, 12
plot.doseResponse (plot.DRtrace), 19
plot.DRtrace, 12, 19

quickInverse, 2, 7, 10, 14, 22, 24, 26, 29
quickIsotone, 2, 4–7, 13–15, 23, 24

slope, 6, 7, 27

wilsonCI, 25, 28