Package 'upndown'

June 7, 2023

Type Package

Title Utilities and Design Aids for Up-and-Down Dose-Finding Studies

Version 0.1.0

Date 2023-06-05

Description
Up-and-Down is the most popular design approach for dose-finding, but has been severely underserved by the statistical computing community. This is the first package to address Up-and-Down's needs. For a recent methodological tutorial on Up-and-Down, see Oron et al. (2022) <doi:10.1097/ALN.0000000000004282>.

License GPL-2

RoxygenNote 7.2.3

Encoding UTF-8

Suggests knitr, rmarkdown

Imports cir, expm, numbers

VignetteBuilder knitr

NeedsCompilation no

Author Assaf P. Oron [cre, aut]

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

Repository CRAN

Date/Publication 2023-06-07 13:30:02 UTC

R topics documented:

`adaptmean` *Up-and-Down averaging estimate with adaptive starting-point*

Description

A dose-averaging estimate based on a concept from Oron (2007). Provides an alternative to reversal-based averaging.

Usage

```
adaptmean(x, maxExclude = 1/3, before = FALSE, full = FALSE)
```

Arguments

`x`	numeric vector: sequence of administered doses, treatments, stimuli, etc.
`maxExclude`	a fraction in $0, 1$ indicating the maximum initial fraction of the vector `x` to exclude from averaging, in case the algorithm-identified starting point occurs too late in the experiment. Default 1/3.
`before`	logical: whether to start the averaging one step earlier than the starting reversal point. Default `FALSE`, and ignored when `all=FALSE`.
`full`	logical: should more detailed information be returned, or only the estimate? (default `FALSE`)

Details

Historically, most up-and-down studies have used dose-averaging estimates. Many of them focus on reversal points either as anchor/cutoff points – points where the averaging begins – or as the **only** doses to use in the estimate. Excluding doses before the anchor/cutoff is done in order to mitigate the bias due to the arbitrary location of the starting dose. The extent of excluded sample depends on the distance between the starting dose and the up-and-down balance point, as well as the random-walk vagaries of an individual experimental run.

Oron (2007) showed that using only reversals and skipping other doses is generally a bad idea, and also noted that a reversal anchor point is not directly tied to the conceptual motivation for having an anchor/cutoff point.

In practice, some *"lucky"* experiments might not need any exclusion at all (because they started right at the balance point), while others might need to exclude dozens of observations. Reversals do not capture this variability well.

The estimation method coded in `adaptmean()` works from a different principle. It identifies **the first crossing point:** the first point at which the dose is *"on the other side"* from the starting point, compared with the average of all remaining doses. The average of all remaining doses is used as a proxy to the (unobservable) balance point. This approach is far closer to capturing the dynamics

described above, and indeed performs well in comparative simulations (Oron et al. 2022, Supplement).

Interestingly, unlike other methods `adaptmean()` does not require the experiment's binary responses as input; only the dose-allocation sequence.

The reason `adaptmean()` has not been further developed nor published, is that like all dose-averaging estimators, at present there doesn't seem to be a reliable confidence interval to accompany any of them.

For UDD target estimation we recommend using centered isotonic regression, a more robust method available together with a confidence interval via `udest`, an up-and-down adapted wrapper to `cir::quickInverse()`. See Oron et al. 2022 (both article and supplement) for further information, as well as the `cir` package vignette.

Value

The point estimate

Author(s)

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

References

- Oron AP. *Up-and-Down and the Percentile-finding Problem.* Ph.D. Dissertation, University of Washington, 2007.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50. See in particular the open-access Supplement.

See Also

- `udest`, the recommended estimation method for up-and-down targets.
- `reversmean` for the commonly-used reversal-anchored averages mentioned in Details.

Examples

```
#'  **An up-and-down experiment that has generated some controversy**
#'
#' Van Elstraete, AC et al. The Median Effective Dose of Preemptive Gabapentin
#'      on Postoperative Morphine Consumption After Posterior Lumbar Spinal Fusion.
#'      *Anesthesia & Analgesia* 2008, 106: 305-308.


# It was a classical median-finding up-and-down study.

doses = c(4:7, 6:13, 12:19, 18:21, 20, 19:23, 22, 21:23, 22:19, 20:23,
          22:24, 23, 22, 23, 22:25, 24:22, rep(23:24,2), 23, 22)
# With U&D, responses (except the last one) can be read off the doses:
responses = c( (1 - sign(diff(doses)))/2, 0 )
```

```
### Let us plot the dose-allocation time series.

# Saving current settings as now required by the CRAN powers-that-be :0
op <- par(no.readonly = TRUE)

par(mar=c(4,4,4,1), mgp=c(2.5,0.8,0), cex.axis = 0.7, las = 1)
udplot(doses, responses, main='Van Elstraete et al. 2008 Study',
       xtitle = "Patient Number", ytitle = 'Gabapentin (mg/kg)')


#' Overlay the ED50 reported in the article (21.7 mg/kg):
abline(h = 21.7)

#' The authors cite a little-known 1991 article by Dixon as the method source.
#' However, in their author rejoinder they claim to have used the Dixon-Mood (1948) estimate.


# Our package does include the Dixon-Mood point estimate.
#  (w/o the CIs, because we do not endorse this estimation approach)
# Does it reproduce the article estimate?
dixonmood(doses, responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(doses, responses), col=2)

# We have found that many articles claiming to use Dixon-Mood (or Dixon-Massey) actually
# Do something else. For example, in this article they report that
#   "it is necessary to reject sequences with three to six identical results".
# Nothing like this appears in the original Dixon-Mood article, where the estimation method
#   involves identifying the less-common response (either 0 or 1), and using only x values
#   associated with these responses; obviating the need to exclude specific sequences.
#
# More generally, these historical estimates have long passed their expiry dates.
#   Their foundation is not nearly as solid as, e.g., linear regression,
#      and it's time to stop using them.

# That said, our package does offer two more types of dose-averaging estimates.
# Both are able to take advantage of the "n+1" dose-allocation, which is determined by
#    the last dose and response:
n = length(doses)
dosePlus1 = doses[n] + ifelse(responses[n]==0, 1, -1)
reversmean(c(doses, dosePlus1), responses)
# Interestingly, in this particular case the answer is very similar to the Dixon-Mood estimate.

# The `reversmean()` default averages all doses from the 3rd reversal point onwards.
# By the way, at what point did the third reversal happen?
#     It'll be the 3rd number in this vector:
reversals(responses)

# Far more commonly in literature, particularly in sensory studies,
#   one encounters the 1960s-era approach (led by Wetherill) of taking *only doses
#  at reversal points, usually starting from the first one. `reversmean()` can do that too:
```

```
wetherill = reversmean(c(doses, dosePlus1), responses, all = FALSE, rstart = 1)
wetherill
# This one gives an even lower result than the previous ones.
abline(h = wetherill, col = 3)

# There's another approach to dose-averaging, although it is not in use anywhere that we know of.
# It does not require the y values at all. The underlying assumption is that the dose
#  sequence has done enough meandering around the true balance point, to provide information
#   about where (approximately) the starting-dose effect is neutralized.
adaptmean(c(doses, dosePlus1))
# Again a bit curiously, this relatively recent approach gives a result similar to what
#   the authors reported (but not similar to the original Dixon-Mood).
# This is not too surprising, since here `adaptmean()` excludes the first one-third of doses,
#  which is approximately what happened if indeed the authors excluded all those long dose-increase
#   sequences at the start.

# All this shows how dicey dose-averaging, at face value a simple and effective method, can become.
# The sample size here is rather large for up-and-down studies, and yet because of the unlucky
#   choice of starting point (which in many studies, due to safety concerns cannot be evaded)
#    there is really no good option of which observations to exclude.

# This is one reason why we strongly recommend using Centered Isotonic Regression as default:
defest = udest(doses, responses, target = 0.5)
abline(h = defest$point, col = 'purple')
# For this dataset, it is the highest of all the estimates.

legend('bottomright', col = c(1:3, 'purple'),
     legend = c("Article's estimate", 'Dixon-Mood', 'Reversals (Wetherill)', 'Standard (CIR)'),
       lty = 1, bty='n', cex = 0.8)

par(op) # Back to business as usual ;)
```

---

bcdmat *Transition Probability Matrices for Up-and-Down Designs*

Description

Transition Probability Matrices for Common Up-and-Down Designs

Usage

```
bcdmat(cdf, target)

classicmat(cdf)

kmatMarg(cdf, k, lowTarget)

kmatFull(cdf, k, lowTarget, fluffup = FALSE)

gudmat(cdf, cohort, lower, upper)
```

Arguments

`cdf`	monotone increasing vector with positive-response probabilities. The number of dose levels M is deduced from vector's length.
`target`	the design's target response rate (`bcdmat()` only).
`k`	the number of consecutive identical responses required for dose transitions (k-in-a-row functions only).
`lowTarget`	logical k-in-a-row functions only: is the design targeting below-median percentiles, with k repeated negative responses needed to level up and only one to level down - or vice versa? Default `FALSE`. See "Details" for more information.
`fluffup`	logical (`kmatFull` only): in the full k-in-a-row internal-state representation, should we *"fluff"* the matrix up so that it has Mk rows and columns (`TRUE`, default), or exclude $k-1$ "phantom" states near one of the boundaries?
`cohort, lower, upper`	`gudmat` only: the cohort (group) size, how many positive responses are allowed for a move upward, and how many are required for a move downward, respectively. For example `cohort=3, lower=0, upper=2` evaluates groups of 3 observations at a time, moves up if none are positive, down if $>= 2$ are positive, and repeats the same dose with 1 positive.

Details

Up-and-Down designs (UDDs) generate random walk behavior, whose theoretical properties can be summarized via a transition probability matrix (TPM). Given the number of doses M, and the value of the cdf F at each dose (i.e., the positive-response probabilities), the specific UDD rules uniquely determine the TPM.

The utilities described here calculate the TPMs of the most common and simplest UDDs:

- The k-in-a-row or **fixed staircase** design common in sensory studies: `kmatMarg(), kmatFull()` (Gezmu, 1996; Oron and Hoff, 2009; see Note). Design parameters are k, a natural number, and whether k negative responses are required for dose transition, or k positive responses. The former is for targets below the median and vice versa.
- The Durham-Flournoy Biased Coin Design: `bcdmat()`. This design can target any percentile via the `target` argument (Durham and Flournoy, 1994).
- The original *"classical"* median-targeting UDD: `classicmat()` (Dixon and Mood, 1948). This is simply a wrapper for `bcdmat()` with `target` set to 0.5.
- Cohort or group UDD: `gudmat()`, with three design parameters for the group size and the up/down rule thresholds (Gezmu and Flournoy, 2006).

Value

An $M \times M$ transition probability matrix, except for `kmatFull()` with $k > 1$ which returns a larger square matrix.

Note

As Gezmu (1996) discovered and Oron and Hoff (2009) further extended, k-in-a-row UDDs with $k > 1$ generate a random walk *with internal states*. Their full TPM is therefore larger than $M \times M$. However, in terms of random-walk behavior, most salient properties are better represented via an $M \times M$ matrix analogous to those of the other designs, with transition probabilities marginalized over internal states using their asymptotic frequencies. This matrix is provided by `kmatMarg()`, while `kmatFull()` returns the full matrix including internal states.

Also, in `kmatFull()` there are two matrix-size options. Near one of the boundaries (upper boundary with `lowTarget = TRUE`, and vice versa), the most extreme k internal states are practically indistinguishable, so in some sense only one of them really exists. Using the `fluffup` argument, users can choose between having a more aesthetically symmetric (but a bit misleading) full $Mk \times Mk$ matrix, or reducing it to its effectivelly true size by removing $k - 1$ rows and columns.

Author(s)

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

References

- Dixon WJ, Mood AM. A method for obtaining and analyzing sensitivity data. *J Am Stat Assoc.* 1948;43:109-126.
- Durham SD, Flournoy N. Random walks for quantile estimation. In: *Statistical Decision Theory and Related Topics V* (West Lafayette, IN, 1992). Springer; 1994:467-476.
- Gezmu M. The Geometric Up-and-Down Design for Allocating Dosage Levels. PhD Thesis. American University; 1996.
- Gezmu M, Flournoy N. Group up-and-down designs for dose-finding. *J Stat Plan Inference.* 2006;136(6):1749-1764.
- Oron AP, Hoff PD. The k-in-a-row up-and-down design, revisited. *Stat Med.* 2009;28:1805-1820.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50.

See Also

- `k2targ`, `ktargOptions` to find the k-in-a-row target-response rate for specific k and vice versa.
- `g2targ`, , `gtargOptions` likewise for group up-and-down.
- `pivec`, `currentvec`, `cumulvec`, which provide probability vectors of dose-allocation distributions using Up-and-Down TPMs.

Examples

```
#  Let's use an 8-dose design, and  a somewhat asymmetric CDF

exampleF = pweibull(1:8, shape = 2, scale = 4)
# You can plot if you want: plot(exampleF)

# Here's how the transition matrix looks for the median-finding classic up-and-down
```

```
round(classicmat(exampleF), 2)
# Note how the only nonzero diagonals are at the opposite corners. That's how
#   odd-n and even-n distributions communicate (see examples for vector functions).
# Also note how "up" probabilities (the 1st upper off-diagnoal) are decreasing,
#   while "down" probabilities (1st lower off-diagonal) are increasing, and
#   start exceeding "up" moves at row 4.

# Now, let's use the same F to target the 90th percentile, which is often
#    the goal of anesthesiology dose-finding studies.
#    We use the biased-coin design (BCD) presented by Durham and Flournoy (1994):

round(bcdmat(exampleF, target = 0.9), 2)

# Note that now there's plenty of probability mass on the diagonal (i.e., repeating same dose).

# Another option, actually with somewhat better operational characteristics,
#   is "k-in-a-row". Let's see what k to use:

ktargOptions(.9, tolerance = 0.05)

# Even though nominally k=7's target is closest to 0.9, it's generally preferable
#    to choose a somewhat smaller k. So let's go with k=6.
# We must also specify whether this is a low (<0.5) or high target.

round(kmatMarg(exampleF, k = 6, lowTarget = FALSE), 2)

# Compare and contrast with the BCD matrix above! At what dose do the "up" and "down"
#   probabilities flip?

# Lastly, if you want to see a 43 x 43 matrix - the full state matrix for k-in-a-row,
#      run the following line:



  round(kmatFull(exampleF, k = 6, lowTarget = FALSE), 2)
```

---

dixonmood *Original Dixon and Mood (1948) point estimate*

Description

Basic version; formula assumes uniform spacing but should work anyway

Usage

```
dixonmood(x, y, full = FALSE, flip = FALSE)
```

Arguments

x	numeric vector: sequence of administered doses, treatments, stimuli, etc.
y	numeric vector: sequence of observed responses. Must be same length as x or shorter by 1, and must be coded TRUE/FALSE or 0/1.
full	logical: should more detailed information be returned, or only the estimate? (default FALSE)
flip	logical: should we flip D-M's approach and use the more-common outcome? (default FALSE)

Details

In their documentation of the Up-and-Down algorithm, Dixon and Mood (1948) presented an estimation method based on tallying and averaging responses, choosing to use only the positive or negative responses (the less-common of the two), since they reasoned the two mirror each other. This is not strictly true: it ignores both leading/trailing "tail" sequences of identical responses, and repeated visits to the boundary dose in case there are dose boundaries.

The Dixon-Mood estimate (sometimes called Dixon-Massey) is provided here mostly for historical reasons and comparative-simulation uses, and also because this estimate is apparently *(and unfortunately)* still in use in some fields. **It should not be used for actual target-dose estimation in real experiments.** It behaves very poorly even under minor deviations from the most optimal conditions (see, e.g., simulations in the supplement to Oron et al. 2022.).

In order to discourage from actual use in experiments, we do not provide a method for the Dixon-Mood estimator's confidence interval, even though the original article did include one. The interval estimate behaves even more poorly than the point estimate.

For UDD target estimation we recommend using centered isotonic regression, available via `udest`, an up-and-down adapted wrapper to `cir::quickInverse()`. See Oron et al. 2022 (both article and supplement) for further information, as well as the `cir` package vignette.

Value

The point estimate

Author(s)

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

References

- Dixon WJ, Mood AM. A method for obtaining and analyzing sensitivity data. *J Am Stat Assoc.* 1948;43:109-126.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50. See in particular the open-access Supplement.

See Also

- udest, the recommended estimation method for up-and-down targets.
- reversmean, The most commonly-used dose-averaging approach (*not* recommended; the recommended one is udest referenced above).

Examples

```
#'  **An up-and-down experiment that has generated some controversy**
#'
#' Van Elstraete, AC et al. The Median Effective Dose of Preemptive Gabapentin
#'      on Postoperative Morphine Consumption After Posterior Lumbar Spinal Fusion.
#'      *Anesthesia & Analgesia* 2008, 106: 305-308.


# It was a classical median-finding up-and-down study.

doses = c(4:7, 6:13, 12:19, 18:21, 20, 19:23, 22, 21:23, 22:19, 20:23,
          22:24, 23, 22, 23, 22:25, 24:22, rep(23:24,2), 23, 22)
# With U&D, responses (except the last one) can be read off the doses:
responses = c( (1 - sign(diff(doses)))/2, 0 )


### Let us plot the dose-allocation time series.

# Saving current settings as now required by the CRAN powers-that-be :0
op <- par(no.readonly = TRUE)

par(mar=c(4,4,4,1), mgp=c(2.5,0.8,0), cex.axis = 0.7, las = 1)
udplot(doses, responses, main='Van Elstraete et al. 2008 Study',
       xtitle = "Patient Number", ytitle = 'Gabapentin (mg/kg)')


#' Overlay the ED50 reported in the article (21.7 mg/kg):
abline(h = 21.7)

#' The authors cite a little-known 1991 article by Dixon as the method source.
#' However, in their author rejoinder they claim to have used the Dixon-Mood (1948) estimate.


# Our package does include the Dixon-Mood point estimate.
#  (w/o the CIs, because we do not endorse this estimation approach)
# Does it reproduce the article estimate?
dixonmood(doses, responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(doses, responses), col=2)

# We have found that many articles claiming to use Dixon-Mood (or Dixon-Massey) actually
# Do something else. For example, in this article they report that
#   "it is necessary to reject sequences with three to six identical results".
# Nothing like this appears in the original Dixon-Mood article, where the estimation method
#   involves identifying the less-common response (either 0 or 1), and using only x values
```

```
#   associated with these responses; obviating the need to exclude specific sequences.
#
# More generally, these historical estimates have long passed their expiry dates.
#   Their foundation is not nearly as solid as, e.g., linear regression,
#      and it's time to stop using them.

# That said, our package does offer two more types of dose-averaging estimates.
# Both are able to take advantage of the "n+1" dose-allocation, which is determined by
#    the last dose and response:
n = length(doses)
dosePlus1 = doses[n] + ifelse(responses[n]==0, 1, -1)
reversmean(c(doses, dosePlus1), responses)
# Interestingly, in this particular case the answer is very similar to the Dixon-Mood estimate.

# The `reversmean()` default averages all doses from the 3rd reversal point onwards.
# By the way, at what point did the third reversal happen?
#     It'll be the 3rd number in this vector:
reversals(responses)

# Far more commonly in literature, particularly in sensory studies,
#   one encounters the 1960s-era approach (led by Wetherill) of taking *only doses
#  at reversal points, usually starting from the first one. `reversmean()` can do that too:
wetherill = reversmean(c(doses, dosePlus1), responses, all = FALSE, rstart = 1)
wetherill
# This one gives an even lower result than the previous ones.
abline(h = wetherill, col = 3)

# There's another approach to dose-averaging, although it is not in use anywhere that we know of.
# It does not require the y values at all. The underlying assumption is that the dose
#  sequence has done enough meandering around the true balance point, to provide information
#   about where (approximately) the starting-dose effect is neutralized.
adaptmean(c(doses, dosePlus1))
# Again a bit curiously, this relatively recent approach gives a result similar to what
#   the authors reported (but not similar to the original Dixon-Mood).
# This is not too surprising, since here `adaptmean()` excludes the first one-third of doses,
#  which is approximately what happened if indeed the authors excluded all those long dose-increase
#   sequences at the start.

# All this shows how dicey dose-averaging, at face value a simple and effective method, can become.
# The sample size here is rather large for up-and-down studies, and yet because of the unlucky
#   choice of starting point (which in many studies, due to safety concerns cannot be evaded)
#    there is really no good option of which observations to exclude.

# This is one reason why we strongly recommend using Centered Isotonic Regression as default:
defest = udest(doses, responses, target = 0.5)
abline(h = defest$point, col = 'purple')
# For this dataset, it is the highest of all the estimates.

legend('bottomright', col = c(1:3, 'purple'),
     legend = c("Article's estimate", 'Dixon-Mood', 'Reversals (Wetherill)', 'Standard (CIR)'),
       lty = 1, bty='n', cex = 0.8)

par(op) # Back to business as usual ;)
```

drplot *Visualizing the dose-response summary of an up-and-down experiment*

Description

Dose-response plotting function for up-and-down data, with doses/stimuli on the x-axis, and the proportion of positive responses on the y-axis. Includes an option to plot the target-dose estimate. Uses utilities from the `cir` package.

Usage

```
drplot(
  x,
  y,
  shape = "X",
  connect = FALSE,
  symbcol = 1,
  percents = FALSE,
  addest = FALSE,
  addcurve = FALSE,
  target = NULL,
  balancePt = target,
  conf = 0.9,
  estcol = "purple",
  estsize = 2,
  estsymb = 19,
  esthick = 2,
  curvecol = "blue",
  ytitle = "Frequency of Positive Response",
  xtitle = "Dose / Stimulus",
  ...
)
```

Arguments

`x`	numeric vector: sequence of administered doses, treatments, stimuli, etc.
`y`	numeric vector: sequence of observed responses. Must be same length as `x`, and must be coded `TRUE/FALSE` or 0/1.
`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 `udplot()` and `FALSE` for `drplot()`.
`symbcol`	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 might be an error message.
`percents`	logical, whether to represent the y-axis as percents rather than a fraction.

addest	logical: should we add the CIR target-dose estimate and its confidence interval? If FALSE (default), then arguments addcurve, target, balancePt, conf, estcol, estsize, estsyml - are all ignored.
addcurve	logical: should we add the complete estimated CIR dose-response curve? Default FALSE, and only relevant when addest = TRUE.
target	The target response rate for which target dose estimate is requested. Must be a single number in $(0, 1)$.
balancePt	In case the design's inherent balance point differs somewhat from target, specify it here to improve estimation accuracy. See Details for further explanation. Otherwise, this argument defaults to be equal to target.
conf	The desired confidence level for the confidence interval. Default 90%. We do not recommend increasing to 95% unless you have ~ 100 or more observations.
estcol, estsize, estsymb, esthick, curvecol	graphical parameters controlling the colors, symbol choice, size, thickness, of the target-dose and CIR-curve visuals.
xtitle, ytitle	x-axis and y-axis titles. Some reasonable defaults are provided, to avoid an annoying error message.
...	Other arguments passed on to plot (e.g., main for the main title).

Details

After an up-and-down experiment, it is highly recommended to plot not just the experiment's *"trace"* time-series (udplot), but also the dose-response summaries. This utility provides a convenient interface for doing that.

- It summarizes the response rates at each participating dose, and plots them. At default, symbol area is proportional to the number of observations at each dose.
- Optionally, the centered-isotonic-regression (CIR) target-dose estimate and its confidence interval are also calculated and plotted.
- A further option allows for plotting the entire estimated CIR dose-response curve.

drplot() is a convenience wrapper to cir::plot.doseResponse, with the added option of plotting the estimate. Some specific options, such as disabling the proportional-area symbol plotting, are accessible only via plot.doseResponse arguments (specified in your drplot() call and passed through the ...).

This is a base-R plot, so you can use additional options, including preceding the plot command with par statements, or following up with legend. When wishing to save to a file, I recommend utilities such as png() or pdf().

Value

Returns invisibly after plotting. If you would like to save the plot to a file, embed the plotting code in a standard R graphics export code sequence, (e.g., pdf(...) before the plotting function, and dev.off() after it).

Author(s)

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

References

- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50.

See Also

- `plot.doseResponse`, `cir` package.
- `udplot` for the "trace" time-series plot.
- `cir` package vignette.

`k2targ` *Up-and-Down Target Calculation and Design Guidance*

Description

Up-and-down target calculation, as well as design options/guidance given a user-desired target.

Usage

```
k2targ(k, lowTarget = FALSE)

ktargOptions(target, tolerance = 0.1, maxk = 20)

g2targ(cohort, lower, upper)

gtargOptions(target, minsize = 2, maxsize = 6, tolerance = 0.1)

bcoin(target, fraction = FALSE, nameplate = FALSE, tolerance = 0.02)
```

Arguments

`k`	the number of consecutive identical responses required for dose transitions (k-in-a-row functions only).
`lowTarget`	logical, `k2targ()` only: is the design targeting below-median percentiles, with k repeated negative responses needed to level up and only one to level down - or vice versa? Default `FALSE`.
`target`	the desired target response rate (as a fraction in $(0, 1)$), where relevant.
`tolerance`	• For `ktargOptions()`, `gtargOptions()`: the half-width of the interval around `target` in which to search for design options. Default 0.1. • For `bcoin()`: the half-width of the interval around 0.5 in which the function recommends to simply use classical UD without a coin, as well as the approximate amount of rounding to the returned coin probability (whether in decimal on rational terms). Default 0.02, and hard-coded to be no less than 0.0001.
`maxk`	`ktargOptions()` only: the maximum value of k to consider.

`cohort, lower, upper`
: `g2targ()` only: the cohort (group) size, how many positive responses are allowed for a move upward, and how many are required for a move downward, respectively. For example `cohort=3, lower=0, upper=2` evaluates groups of 3 observations at a time, moves up if none are positive, down if $>= 2$ are positive, and repeats the same dose with 1 positive.

`minsize, maxsize`
: `gtargOptions()` only: the minimum and maximum cohort size to consider. `minsize` has to be at least 2 (cohort size 1 is equivalent to classical UD).

`fraction`
: `bcoin()` only: whether to report the coin probability as a rational rather than decimal fraction. Default `FALSE`.

`nameplate`
: `bcoin()` only: in case `fraction = TRUE`, whether to return the "exact" rational probability, or allow some nudging of the resulting balance point towards the median. Default `FALSE`, and moot when `fraction = FALSE`.

Details

This suite of utilities helps users

- Figure out the approximate target response-rate (a.k.a. the *balance point*), given design parameters;
- Suggest potential design parameters, given user's desired target response-rate and other constraints.

Up-and-down designs (UDDs) generate random walks over dose space, with most dose-allocations usually taking place near the design's de-facto target percentile, called the **"balance point"** by some theorists to distinguish it from the user-designated target in case they differ (Oron and Hoff 2009, Oron et al. 2022).

Most k-in-a-row and group UDD parameter combinations yield balance points that are irrational percentiles of the dose-response function, and therefore are unappealing as official experimental targets.

However, since the UD dose distribution has some width, and since even the balance point itself is only a close approximation for the actual average of allocated doses, the user's target **does not have to be identical to the balance point.** It only needs to be *"close enough"*.

The `k2targ()` and `g2targ()` utilities are intended for users who already have a specific k-in-a-row or group design in mind, and only want to verify its balance point. The complementary utilities `ktargOptions(), gtargOptions()` provide a broader survey of design-parameter options within user-specified constraints, given a desired target.

Lastly, `bcoin()` returns the biased-coin probabilities given the user's designated target. In contrast to the two other UDDs described above, the biased-coin design can target any percentile with a precisely matched balance point. That said, k-in-a-row and group UDDs offer some advantages over biased-coin in terms of properties and operational simplicity.

`bcoin()` can return the probability as a decimal (default) or approximate rational fraction. In the latter case, if `nameplate` is set to `TRUE`, you will get the exact *"nameplate"* coin probability $\Gamma/(1-\Gamma)$, with Γ being the target percentile between 0 and 1. However, the default `nameplate = FALSE` might nudge the coin to yield a balance point somewhat closer to the median. This choice is based upon the theoretical finding that the biased-coin design does tend to concentrate doses a bit further

away from the median than the balance point would suggest (Oron and Hoff, 2009). See more information in `bcoin()`'s argument descriptions.

Value

- `k2targ(), g2targ()`: the official balance point given the user-provided design parameters.
- `ktargOptions(), gtargOptions()`: a `data.frame` with design parameters and official balance point, for all options that meet user-provided constraints. A printed string provides dose transition rule guidance.
- `bcoin()`: a printed string that informs user of the biased-coin design rules, including the 'coin' probability in its user-chosen format (decimal or fraction). In case the user-desired target is 0.5 or very close to it, the string will inform user that they are better off just using classical UDD without a coin.

Author(s)

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

References

- Durham SD, Flournoy N. Random walks for quantile estimation. In: *Statistical Decision Theory and Related Topics V* (West Lafayette, IN, 1992). Springer; 1994:467-476.
- Gezmu M, Flournoy N. Group up-and-down designs for dose-finding. *J Stat Plan Inference.* 2006;136(6):1749-1764.
- Oron AP, Hoff PD. The k-in-a-row up-and-down design, revisited. *Stat Med.* 2009;28:1805-1820.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50.

See Also

- `bcdmat` for the functions calculating transition probability matrices for various up-and-down designs.
- `pivec` for functions calculating key probability vectors for the designs.

`pivec` *Key Probability Vectors of Up-and-Down Designs*

Description

Dose-allocation probability vectors that quantify the instantaneous, cumulative, and asymptotic behavior of Up-and-Down designs.

Usage

```
pivec(cdf, matfun, ...)

currentvec(cdf, matfun, n, startdose = NULL, ...)

cumulvec(
  cdf,
  matfun,
  n,
  startdose = NULL,
  proportions = TRUE,
  exclude = 0,
  ...
)
```

Arguments

`cdf`	monotone increasing vector with positive-response probabilities. The number of dose levels M is deduced from vector's length.
`matfun`	The function to calculate the TPM. Depends on the specific design; see `bcdmat`. For all functions except `classicmat`, user must provide auxiliary parameters via `...`.
`...`	Arguments passed on to the design's matrix-calculating function.
`n`	For `currentvec`, `cumulvec`, at what step (= after how many observations) in the experiment would you like the vector calculated?
`startdose`	For `currentvec`, `cumulvec`, where does the experiment start? To be given as a dose-level index between 1 and M. If left as `NULL` (default), function will assume the equivalent of *"fair die roll"* among all doses. User can also specify your own M-length probability vector.
`proportions`	Logical (`cumulvec` only) Would you like the results returned as proportions (= a probability vector; `TRUE`, default), or as cumulative allocation counts?
`exclude`	Integer (`cumulvec` only) Should the cumulative distribution exclude a certain number of initial observations? Default 0.

Details

Up-and-Down designs (UDDs) generate random walk behavior, which concentrates doses around the target quantile. Asymptotically, dose allocations follow a stationary distribution $\boldsymbol{\pi}$ which can be calculated given the number of doses M, and the value of the cdf F at each dose (i.e., the positive-response probabilities), and the specific UDD rules. No matter the starting dose, the allocation distribution converges to $\boldsymbol{\pi}$ at a geometric rate (Diaconis and Stroock, 1991).

Three functions are offered:

- `pivec()` returns $\boldsymbol{\pi}$.
- `currentvec()` returns the current (instantaneous) allocation distribution at step `n`, using the formula from Diaconis and Stroock (1991).

- `cumulvec()` returns the *cumulative* allocations, i.e., the expected proportions (or counts) of allocations during the experiment after `n` observations. This function is perhaps of greatest practical use.

All functions first calculate the transition probability matrix (TPM), by calling one of the functions described under `bcdmat`. See that help page for more details.

Value

A vector of allocation frequencies/probabilities for the doses, summing up to 1. Exception: `cumulvec(propotions = FALSE)` returns a vector of expected allocation counts, summing up to `n - exclude`.

Note

When using the k-in-a-row design, set `matfun = kmatMarg`, not `kmatFull`.

Author(s)

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

References

- Diaconis P, Stroock D. Geometric Bounds for Eigenvalues of Markov Chains. *Ann. Appl. Probab.* 1991;1(1):36-61.
- Hughes BD. *Random Walks and Random Environments, Vol. 1.* Oxford University Press, 1995.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50.

See Also

- `bcdmat` for the functions calculating transition probability matrices for various up-and-down designs.
- `k2targ` for target-finding design aids.

Examples

```
#----- Classical UD Example -----#

# An example used in Oron et al. 2022, Fig. 2.
# It is presented here via the original motivating story:
# "Ketofol"  is a commonly-used anesthesia-inducing mix known to combine its 2 components'
# beneficial properties, while each component mitigates the other's harmful side-effects.
# In particular:
#     Propofol reduces blood pressure while ketamine raises it.
# What is *not* known at present, is which mix proportions produce
# 0 "delta-BP" on average among the population.

# The classical UD design below administers the mix 0-100% ketamine in 10% increments.
#    The design will concentrate doses around the point where half the population
```

```
#    experiences 0 "delta-BP". (the 'zeroPt' parameter in the code)

doses = seq(0, 100, 10)
m=length(doses) # 11 dose levels

zeroPt=63 # the zero-BP point, in units of percent ketamine

# We assume a Normal ("Probit") dose-response curve,
#   and calculate the value of F (i.e.,  prob (delta BP > 0) at the doses:
equivF = pnorm( (doses - zeroPt) / 20)
round(equivF, 3)

# The vector below represents the values feeding into the Fig. 2B barplot.
# "startdose = 6" means the experiment begins from the 6th out of 11 doses, i.e., a 50:50 mix.


  round(cumulvec(cdf = equivF, matfun = classicmat, startdose = 6, n = 30), 3)

# Compare with the *instantaneous* probability distribution to the 30th patient:

round(currentvec(cdf = equivF, matfun = classicmat, startdose = 6, n = 30), 3)
# Classic up-and-down has quasi-periodic behavior with a (quasi-)period of 2.
# Compare the instantaneous vectors at n=30 and 29:
round(currentvec(cdf = equivF, matfun = classicmat, startdose = 6, n = 29), 3)
# Note the alternating near-zero values. Distributions at even/odd n "communicate"
#    with each other only via the dose boundaries.

# Lastly, the asymptotic/stationary distribution. Notice there is no 'n' argument.

round(pivec(cdf = equivF, matfun = classicmat), 3)

# The cumulative vector at n=30 is not very far from the asymptotic vector.
# The main difference is that at n=30 there's still a bit more
#    probability weight at the starting dose.
# We can check how much of that extra weight is from the 1st patient, by excluding that data point:

round(cumulvec(cdf = equivF, matfun = classicmat, startdose = 6, n = 30, exclude = 1), 3)
```

`reversmean` *Reversal-anchored averaging estimators for Up-and-Down*

Description

Dose-averaging target estimation for Up-and-Down experiments, historically the most popular approach, but not recommended as primary nowadays. Provided for completeness.

Usage

```
reversmean(
  x,
  y,
  rstart = 3,
  all = TRUE,
  before = FALSE,
  maxExclude = 1/3,
  full = FALSE
)

reversals(y)
```

Arguments

x	numeric vector: sequence of administered doses, treatments, stimuli, etc.
y	numeric vector: sequence of observed responses. Must be same length as x or shorter by 1, and must be coded TRUE/FALSE or 0/1.
rstart	the reversal point from which the averaging begins. Default 3, considered a good compromise between performance and robustness. See Details.
all	logical: from the starting point onwards, should all values of x be used (TRUE, default), or only reversal points as in the Wetherill et al. approach?
before	logical: whether to start the averaging one step earlier than the starting reversal point. Default FALSE, and ignored when all=FALSE.
maxExclude	a fraction in $0, 1$ indicating the maximum initial fraction of the vector x to exclude from averaging, in case the algorithm-identified starting point occurs too late in the experiment. Default 1/3.
full	logical: should more detailed information be returned, or only the estimate? (default FALSE)

Details

Up-and-Down designs (UDDs) allocate doses in a random walk centered nearly symmetrically around a balance point. Therefore, a modified average of allocated doses could be a plausible estimate of the balance point's location.

During UDDs' first generation, a variety of dose-averaging estimators was developed, with the one proposed by Wetherill et al. (1966) eventually becoming the most popular. This estimator uses only doses observed at *reversal* points: points with a negative response following a positive one, or vice versa. More recent research (Kershaw 1985, 1987; Oron et al. 2022, supplement) strongly indicates that in fact it is better to use all doses starting from some cut-point, rather than skip most of them and choose only reversals.

The `reversals()` utility identifies reversal points, whereas `reversmean()` produces a dose-averaging estimate whose starting cut-point is determined by a reversal. User can choose whether to use all doses from that cut-point onwards, or only the reversals as in the older approaches. A few additional options make the estimate even more flexible.

More broadly, dose-averaging despite some advantages is not very robust, and also **lacks an interval estimate with reliable coverage.** Therefore, `reversmean()` provides neither a confidence interval nor a standard error.

For UDD target estimation we recommend using centered isotonic regression, available via `udest`, an up-and-down adapted wrapper to `cir::quickInverse()`. See Oron et al. 2022 (both article and supplement) for further information, as well as the `cir` package vignette.

Value

For `reversals()`, the indices of reversal points. For `reversmean()`, if `full=FALSE` returns the point estimate and otherwise returns a data frame with the estimate as well, as the index of the cutoff point used to start the averaging.

Author(s)

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

References

- Kershaw CD: A comparison of the estimators of the ED50 in up-and-down experiments. *J Stat Comput Simul* 1987; 27:175–84.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50. See in particular the open-access Supplement.
- Wetherill GB, Chen H, Vasudeva RB: Sequential estimation of quantal response curves: A new method of estimation. *Biometrika* 1966; 53:439–54

See Also

- `udest`, the recommended estimation method for up-and-down targets.
- `adaptmean`, an unpublished but arguably better approach to dose-averaging (this is *not* the recommended method though; that would be `udest` referenced above).

Examples

```
#'  **An up-and-down experiment that has generated some controversy**
#'
#' Van Elstraete, AC et al. The Median Effective Dose of Preemptive Gabapentin
#'      on Postoperative Morphine Consumption After Posterior Lumbar Spinal Fusion.
#'      *Anesthesia & Analgesia* 2008, 106: 305-308.


# It was a classical median-finding up-and-down study.

doses = c(4:7, 6:13, 12:19, 18:21, 20, 19:23, 22, 21:23, 22:19, 20:23,
          22:24, 23, 22, 23, 22:25, 24:22, rep(23:24,2), 23, 22)
# With U&D, responses (except the last one) can be read off the doses:
responses = c( (1 - sign(diff(doses)))/2, 0 )
```

```
### Let us plot the dose-allocation time series.

# Saving current settings as now required by the CRAN powers-that-be :0
op <- par(no.readonly = TRUE)

par(mar=c(4,4,4,1), mgp=c(2.5,0.8,0), cex.axis = 0.7, las = 1)
udplot(doses, responses, main='Van Elstraete et al. 2008 Study',
       xtitle = "Patient Number", ytitle = 'Gabapentin (mg/kg)')


#' Overlay the ED50 reported in the article (21.7 mg/kg):
abline(h = 21.7)

#' The authors cite a little-known 1991 article by Dixon as the method source.
#' However, in their author rejoinder they claim to have used the Dixon-Mood (1948) estimate.


# Our package does include the Dixon-Mood point estimate.
#  (w/o the CIs, because we do not endorse this estimation approach)
# Does it reproduce the article estimate?
dixonmood(doses, responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(doses, responses), col=2)

# We have found that many articles claiming to use Dixon-Mood (or Dixon-Massey) actually
# Do something else. For example, in this article they report that
#   "it is necessary to reject sequences with three to six identical results".
# Nothing like this appears in the original Dixon-Mood article, where the estimation method
#   involves identifying the less-common response (either 0 or 1), and using only x values
#   associated with these responses; obviating the need to exclude specific sequences.
#
# More generally, these historical estimates have long passed their expiry dates.
#   Their foundation is not nearly as solid as, e.g., linear regression,
#      and it's time to stop using them.

# That said, our package does offer two more types of dose-averaging estimates.
# Both are able to take advantage of the "n+1" dose-allocation, which is determined by
#    the last dose and response:
n = length(doses)
dosePlus1 = doses[n] + ifelse(responses[n]==0, 1, -1)
reversmean(c(doses, dosePlus1), responses)
# Interestingly, in this particular case the answer is very similar to the Dixon-Mood estimate.

# The `reversmean()` default averages all doses from the 3rd reversal point onwards.
# By the way, at what point did the third reversal happen?
#     It'll be the 3rd number in this vector:
reversals(responses)

# Far more commonly in literature, particularly in sensory studies,
#   one encounters the 1960s-era approach (led by Wetherill) of taking *only doses
#  at reversal points, usually starting from the first one. `reversmean()` can do that too:
wetherill = reversmean(c(doses, dosePlus1), responses, all = FALSE, rstart = 1)
```

```
wetherill
# This one gives an even lower result than the previous ones.
abline(h = wetherill, col = 3)

# There's another approach to dose-averaging, although it is not in use anywhere that we know of.
# It does not require the y values at all. The underlying assumption is that the dose
#  sequence has done enough meandering around the true balance point, to provide information
#   about where (approximately) the starting-dose effect is neutralized.
adaptmean(c(doses, dosePlus1))
# Again a bit curiously, this relatively recent approach gives a result similar to what
#   the authors reported (but not similar to the original Dixon-Mood).
# This is not too surprising, since here `adaptmean()` excludes the first one-third of doses,
#  which is approximately what happened if indeed the authors excluded all those long dose-increase
#   sequences at the start.

# All this shows how dicey dose-averaging, at face value a simple and effective method, can become.
# The sample size here is rather large for up-and-down studies, and yet because of the unlucky
#   choice of starting point (which in many studies, due to safety concerns cannot be evaded)
#    there is really no good option of which observations to exclude.

# This is one reason why we strongly recommend using Centered Isotonic Regression as default:
defest = udest(doses, responses, target = 0.5)
abline(h = defest$point, col = 'purple')
# For this dataset, it is the highest of all the estimates.

legend('bottomright', col = c(1:3, 'purple'),
     legend = c("Article's estimate", 'Dixon-Mood', 'Reversals (Wetherill)', 'Standard (CIR)'),
       lty = 1, bty='n', cex = 0.8)

par(op) # Back to business as usual ;)
```

`udest` *Centered-Isotonic-Regression (CIR) Estimate for the Up-and-Down Target Dose*

Description

Centered Isotonic Regression (CIR) is an extension of isotonic regression (IR), substantially improving upon IR's estimation performance in the dose-response and dose-finding contexts (Oron and Flournoy 2017, Flournoy and Oron 2020). CIR is the recommended method for estimating up-and-down targets.

Usage

```
udest(x, y, target, balancePt = target, conf = 0.9, ...)
```

Arguments

`x`	numeric vector: sequence of administered doses, treatments, stimuli, etc.

`y`	numeric vector: sequence of observed responses. Must be same length as `x`, and must be coded `TRUE/FALSE` or 0/1.
`target`	The target response rate for which target dose estimate is requested. Must be a single number in $(0, 1)$.
`balancePt`	In case the design's inherent balance point differs somewhat from `target`, specify it here to improve estimation accuracy. See Details for further explanation. Otherwise, this argument defaults to be equal to `target`.
`conf`	The desired confidence level for the confidence interval. Default 90%. We do not recommend increasing to 95% unless you have ~ 100 or more observations.
`...`	Pass-through argument added for flexible calling context.

Details

CIR and related methods are available in the `cir` package. The `udest()` function in the present package provides a convenient wrapper for `cir::quickInverse()`, with arguments already set to the appropriate values for estimating the target dose after an up-and-down experiment. The function also returns a confidence interval as default.

WARNING! You should not estimate target doses too far removed from the design's actual balance point (definitely no further than 0.1, e.g., estimating the 33rd percentile for a design whose balance point is the median). As Flournoy and Oron (2020) explain, observed response rates are biased away from the balance point. Even though `udest()` performs the rudimentary bias correction described in that article, practically speaking this correction's role is mostly to expand the confidence intervals in response to the bias. It cannot guarantee to provide reliable off-balance-point estimates.

Value

A one-row data frame with 4 variables: `target`, `point` (the point estimate), `lowerXYconf, upperXYconf` (the confidence bounds, with `XY` standing for the percents, default `90`).

Author(s)

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

References

- Oron AP, Flournoy N. Centered Isotonic Regression: Point and Interval Estimation for Dose-Response Studies. *Statistics in Biopharmaceutical Research* 2017; 9, 258-267. Author's public version available on arxiv.org.
- Flournoy N, Oron AP. Bias Induced by Adaptive Dose-Finding Designs. *Journal of Applied Statistics* 2020; 47, 2431-2442.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50. See in particular the open-access Supplement.

See Also

- `quickInverse`, `cir` package.
- `cir` package vignette.

Examples

```
#'  **An up-and-down experiment that has generated some controversy**
#'
#' Van Elstraete, AC et al. The Median Effective Dose of Preemptive Gabapentin
#'      on Postoperative Morphine Consumption After Posterior Lumbar Spinal Fusion.
#'      *Anesthesia & Analgesia* 2008, 106: 305-308.

# It was a classical median-finding up-and-down study.

doses = c(4:7, 6:13, 12:19, 18:21, 20, 19:23, 22, 21:23, 22:19, 20:23,
          22:24, 23, 22, 23, 22:25, 24:22, rep(23:24,2), 23, 22)
# With U&D, responses (except the last one) can be read off the doses:
responses = c( (1 - sign(diff(doses)))/2, 0 )


#' ### Plots plots plots!

# Saving current settings as now required by the CRAN powers-that-be :0
op <- par(no.readonly = TRUE)

layout(t(1:2), widths=3:2)
par(mar=c(4,4,4,1), mgp=c(2.5,0.8,0), cex.axis = 0.7, las = 1)

#' The experimental trajectory / time-series / "trace" (pick your favorite name!)
#' Note the changed argument names for x and y axis titles
udplot(doses, responses, main='',
        xtitle = "Patient Number", ytitle = 'Gabapentin (mg/kg)')
#' Compare with the article's Figure 1; the line below makes it look more similar
udplot(doses, responses, shape='square', connect=TRUE)

# The dose-response plot, rarely encountered in U&D articles.
# We can also add the CIR estimate right there:
drplot(doses, responses, main=' Dose-Response', percents = TRUE,
       addest = TRUE, target = 0.5, addcurve = TRUE,
       xtitle = 'Gabapentin (mg/kg)', ytitle = "Percent Effective")

#' ### Estimates

#' Let us actually see the numbers of those Centered-Isotonic-Regression (CIR) estimates!
#' Note that our default confidence-interval is 90%. Change it via the 'conf' argument.

udest(doses, responses, target = 0.5)
#' Compare with the article: 21.7 mg/kg (95% CI 19.923.5).
#' They cite a little-known 1991 article by Dixon as the method source.
#' However, in their author rejoinder they claim to have used the Dixon-Mood estimate.
#'
#' ## Toy example of plotting a group UD dataset
#'
#' Also showing off some udplot() options
#'
#' Not an actual experiment (made-up data)
```

```
#' The design is purportedly GUD (3,0,1), targeting the 20th percentile
#'

gsize = 3
x = rep(c(1:3, 2:4), each = gsize)
y = c(rep(0, 8), 1, rep(0,7), 1, 1)

udplot(x=x, y=y, cohort=gsize, connect=FALSE, shape='triangle')

par(op) # Back to business as usual ;)
```

---

udplot *Visualizing the time series of an up-and-down experiment*

Description

Plotting function for the "trace" (time series) of an up-and-down experiment, showing the observation order on the x-axis, and the dose *(treatment, stimulus, etc.)* strength on the y-axis. Uses utilities from the `cir` package.

Usage

```
udplot(
  x,
  y,
  cohort = NULL,
  shape = "circle",
  connect = TRUE,
  symbcol = 1,
  doselabels = NULL,
  xtitle = "Observation Order",
  ytitle = "Dose / Stimulus",
  ...
)
```

Arguments

`x`	numeric vector: sequence of administered doses, treatments, stimuli, etc.
`y`	numeric vector: sequence of observed responses. Must be same length as `x`, and must be coded `TRUE/FALSE` or 0/1.
`cohort`	for a group/cohort UD design, the cohort/group size (a single number). In case of variable cohort size, this can be a vector the same length as `x, y`, with each observation's cohort assignment.
`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 `udplot()` and `FALSE` for `drplot()`.

`symbcol`	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 might be an error message.
`doselabels`	(`DRtrace` only) Dose values to be plotted along the y-axis. If `NULL` (default), those will be the doses in the dataset (i.e., `sort(unique(x))`).
`xtitle, ytitle`	x-axis and y-axis titles. Some reasonable defaults are provided, to avoid an annoying error message.
`...`	Other arguments passed on to `plot` (e.g., `main` for the main title).

Details

This simple and handy visualization approach was presented already by Dixon and Mood (1948).

- It conveys directly the meaning of *"up-and-down"*, because the administered dose/stimulus strength is on the y-axis, whereas observation order is on the x-axis.
- Filled symbols stand for positive responses and open symbols for negative.
- The design's transition rules can be usually inferred directly from the plot.

`udplot()` is a convenience wrapper to `cir::plot.DRtrace`. This is a base-R plot, so you can use additional options, including preceding the plot command with `par` statements, or following up with `legend`. When wishing to save to a file, I recommend utilities such as `png()` or `pdf()`.

Value

Returns invisibly after plotting. If you would like to save the plot to a file, embed the plotting code in a standard R graphics export code sequence, (e.g., `pdf(...)` before the plotting function, and `dev.off()` after it).

Author(s)

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

See Also

- `plot.DRtrace`, `cir` package.
- `drplot` for the up-and-down dose-response and estimate plotting.
- `cir` package vignette.

#' @references

- Dixon WJ, Mood AM. A method for obtaining and analyzing sensitivity data. *J Am Stat Assoc.* 1948;43:109-126.
- Oron AP, Souter MJ, Flournoy N. Understanding Research Methods: Up-and-down Designs for Dose-finding. *Anesthesiology* 2022; 137:137–50.

Examples

```
#'  **An up-and-down experiment that has generated some controversy**
#'
#' Van Elstraete, AC et al. The Median Effective Dose of Preemptive Gabapentin
#'      on Postoperative Morphine Consumption After Posterior Lumbar Spinal Fusion.
#'      *Anesthesia & Analgesia* 2008, 106: 305-308.

# It was a classical median-finding up-and-down study.

doses = c(4:7, 6:13, 12:19, 18:21, 20, 19:23, 22, 21:23, 22:19, 20:23,
          22:24, 23, 22, 23, 22:25, 24:22, rep(23:24,2), 23, 22)
# With U&D, responses (except the last one) can be read off the doses:
responses = c( (1 - sign(diff(doses)))/2, 0 )


#' ### Plots plots plots!

# Saving current settings as now required by the CRAN powers-that-be :0
op <- par(no.readonly = TRUE)

layout(t(1:2), widths=3:2)
par(mar=c(4,4,4,1), mgp=c(2.5,0.8,0), cex.axis = 0.7, las = 1)

#' The experimental trajectory / time-series / "trace" (pick your favorite name!)
#' Note the changed argument names for x and y axis titles
udplot(doses, responses, main='',
        xtitle = "Patient Number", ytitle = 'Gabapentin (mg/kg)')
#' Compare with the article's Figure 1; the line below makes it look more similar
udplot(doses, responses, shape='square', connect=TRUE)

# The dose-response plot, rarely encountered in U&D articles.
# We can also add the CIR estimate right there:
drplot(doses, responses, main=' Dose-Response', percents = TRUE,
       addest = TRUE, target = 0.5, addcurve = TRUE,
       xtitle = 'Gabapentin (mg/kg)', ytitle = "Percent Effective")

#' ### Estimates

#' Let us actually see the numbers of those Centered-Isotonic-Regression (CIR) estimates!
#' Note that our default confidence-interval is 90%. Change it via the 'conf' argument.

udest(doses, responses, target = 0.5)
#' Compare with the article: 21.7 mg/kg (95% CI 19.923.5).
#' They cite a little-known 1991 article by Dixon as the method source.
#' However, in their author rejoinder they claim to have used the Dixon-Mood estimate.
#'
#' ## Toy example of plotting a group UD dataset
#'
#' Also showing off some udplot() options
#'
#' Not an actual experiment (made-up data)
```

```
#' The design is purportedly GUD (3,0,1), targeting the 20th percentile
#'

gsize = 3
x = rep(c(1:3, 2:4), each = gsize)
y = c(rep(0, 8), 1, rep(0,7), 1, 1)

udplot(x=x, y=y, cohort=gsize, connect=FALSE, shape='triangle')

par(op) # Back to business as usual ;)
```

---

validUDinput *Data Validation Utilities for* upndown

Description

Validation of input values

Usage

```
validUDinput(cdf, target)

checkTarget(target, tname = "Target")

checkCDF(cdf)

checkNatural(k, parname, toolarge = 1000)

checkDose(x, maxfrac = 0.5)

checkResponse(y)
```

Arguments

cdf	vector of values, should be nondecreasing between 0 and 1 (inclusive)
target	numeric value(s), should be between 0 and 1 (exclusive)
k	(checkNatural() only) input number to check whether it's a natural number
parname, tname	string, name of variable to plug in for reporting the error back
toolarge	(checkNatural() only) what number would be considered too large to be realistic?
x	(checkDose() only) input object to be verified as valid dose values
maxfrac	(checkDose() only) maximum number of unique values (as fraction of sample size) considered realistic for up-and-down data. Default one-half.
y	(checkResponse() only) input object to be verified as valid response values ('TRUE/FALSE or 0/1)

Value

If a validation issue is found, these functions stop with a relevant error message. If no issue is found, they run through without returning a value.

Index