Package ‘upndown’

June 16, 2025

Type Package

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

Version 0.3.0

Date 2025-06-15

Description Up-and-Down (UD) is the most popular design approach for dose-finding, but it has been severely under-served by the statistical and computing communities. This is the first package that comprehensively addresses UD's needs. Recent applied UD tutorial: Oron et al., 2022 <doi:10.1097/ALN.0000000000004282>. Recent methodological overview: Oron and Flournoy, 2024 <doi:10.51387/24-NEJSDS74>.

License GPL-2

RoxygenNote 7.3.2

Encoding UTF-8

Suggests knitr, rmarkdown

Imports cir, expm, MASS, plyr

VignetteBuilder knitr

NeedsCompilation no

Author Assaf P. Oron [cre, aut]

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

Repository CRAN

Date/Publication 2025-06-16 10:20:05 UTC

Contents

bcdmat 2
dfboot 5
dfsim 7
dixonmood 10
drplot 14
dynamean 16
k2targ 21
krow 23

pivec 24
reversmean 27
udest 32
udplot 35
validUDinput 38

Index **40**

`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`), or exclude $k-1$ "phantom" states near the less-likely-to-be-visited boundary (`FALSE`, default)?
`cohort`	`gudmat` only: the cohort (group) size
`lower`, `upper`	(`gudmat` only) 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 `lowTarget` which determines whether `k` positive responses are required for dose transition, or `k` negative 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).
- Cohort or group UDD: `gudmat()`, with three design parameters `cohort, lower, upper`, for the group size and the up/down rule thresholds (Gezmu and Flournoy, 2006).
- The original *"classical"* median-targeting UDD: `classicmat()` (Dixon and Mood, 1948). The classical UDD can be framed as a special case of each of the other 3 UDDs. Functionally, this utility is simply a wrapper for `bcdmat()` with `target` set to 0.5.

Value

An $M \times M$ transition probability matrix, except for `kmatFull()` with $k > 1$ which returns a larger square matrix; see Note below for details on the latter.

Note

As Gezmu (1996) discovered and Oron and Hoff (2009) further extended, k-in-a-row UDDs with $k > 1$ generate what can be described as a random walk **with `k' internal states**. Their full TPM is therefore la`
Marg(), while kmatFull()` returns the full matrix including internal states. Another perspective on this intriguing design, viewing it as generaating a *semi-Markov process*, with equivalent results (Sada Allo et al., in review).

In `kmatFull()`, there are two matrix-size options. At one of the boundary dose-levels (upper boundary with `lowTarget = TRUE`, and vice versa), the k internal states are practically indistinguishable, so arguably only one of them exists. Hence, the most compact TPM representation, and the function default, is $[(M-1)k+1] \times [(M-1)k+1]$. Using `fluffup = TRUE`, users can choose a more aesthetically symmetric (but a bit misleading) full $Mk \times Mk$ matrix.

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, Flournoy N. Up-and-Down: The Most Popular, Most Reliable, and Most Overlooked Dose-Finding Design. *New Eng J Stat Data Science* 2024; 1-12.
- 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.
- Sada M, Flournoy N, Oron AP, Moler J. The K-in-a-row design as a semi-Markov process. Preprint, University of Navarra.

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)
```

---

dfboot *Generic percentile dose-response / dose-finding bootstrap routine*

Description

Bootstrap routine for resampling a dose-finding or dose-response experiment. The bootstrap replicates are generated from a centered-isotonic-regression (CIR) estimate of the dose-response function, rather than resampled directly.

Usage

```
dfboot(
  x,
  y,
  doses = NULL,
  estfun = dynamean,
  design,
  desArgs,
  target,
  balancePt = target,
  conf = 0.9,
  B = 1000,
  seed = NULL,
  randstart = TRUE,
  showdots = TRUE,
  full = 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.
`doses`	the complete set of dose values that *could* have been included in the experiment. Must include all unique values in `x`.
`estfun`	the estimation function to be bootstrapped. Default `dynamean`
`design, desArgs`	design details passed on to `dfsim`; the former is a function and the latter a list of its arguments and values. For self-consistent bootstrapping, this must specify the design used in the actual experiment. See `dfsim`.
`target`	The target percentile to be estimated (as a fraction). Again must be the same one estimated in the actual experiment. Default 0.5.
`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 CI's confidence level, as a fraction in (0,1).
`B`	Size of bootstrap ensemble, default 1000.
`seed`	Random seed; default `NULL` which leads to a "floating" seed, varying between calls.
`randstart`	Logical: should the bootstrap runs randomize the starting dose, or use the same starting dose as the actual experiment? Default `TRUE`, which we expect to produce better properties. The randomization will be weighted by the real data's dose-specific sample sizes.
`showdots`	Logical: should "progress dots" be printed out as the bootstrap runs progress? Default `TRUE`
`full`	Logical: controls how detailed the output is. Default (`FALSE`) is only the resulting interval bounds, while `TRUE` returns a list with the full bootstrap ensemble of doses, responses and estimates, as well as the generating dose-response curve and the bootstrap's dose set.
`...`	Additional parameters passed on to estimation functions.

Details

The function should be able to generate bootstrap resamples of any dose-finding design, as long as `design, desArgs` are specified correctly. For the "Classical" median-finding UDD, use `design = krow, desArgs = list(k=`
For other UDDs, see `dfsim`.

Like Chao and Fuh (2001) and Stylianou et al. (2003), the bootstrap samples are generated indirectly, by estimating a dose-response curve F from the data, then generating an ensemble of bootstrap experiments using the same design used in the original experiment. Unlike these two which used parametric or isotonic regression, respectively, with no bias-mitigation and no additional provisions to improve coverage, our implementation uses CIR with the Flournoy and Oron (2020) bias-mitigation. When feasible, it also allows the bootstrap runs to extend up to 2 dose-levels in each direction, beyond the doses visited in the actual experiment.

Note

This function can be run stand-alone, but it is mostly meant to be called in the backend, in case a dose-averaging estimate "wants" a confidence interval (which is default behavior for `dynamean(), reversmean()` at present). You are welcome to figure out how to run it stand-alone, but I do not provide example code here since we still recommend CIR and its analytically-informed intervals over dose-averaging with bootstrap intervals. If you would like to run general up-and-down or dose-finding simulations, see `dfsim()` and its code example.

Author(s)

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

References

- Chao MT, Fuh CD. Bootstrap methods for the up and down test on pyrotechnics sensitivity analysis. Statistica Sinica. 2001 Jan 1:1-21.
- Flournoy N, Oron AP. Bias induced by adaptive dose-finding designs. J Appl Stat. 2020;47(13-15):2431-2442.
- Stylianou M, Proschan M, Flournoy N. Estimating the probability of toxicity at the target dose following an up-and-down design. Statistics in medicine. 2003 Feb 28;22(4):535-43.

See Also

`dfsim`

`dfsim` *Generalized Dose-Finding Ensemble Simulator*

Description

This function simulates sequential dose-finding experiments on a fixed dose grid. The response function is (implicitly) assumed monotone in `(0,1)`

Usage

```
dfsim(
  n,
  starting = NULL,
  sprobs = NULL,
  cohort = 1,
  Fvals,
  ensemble = dim(Fvals)[2],
  design = krow,
  desArgs = list(k = 1),
  thresholds = NULL,
  seed = NULL,
  showdots = TRUE
)
```

Arguments

`n`	sample size
`starting`	the starting dose level. If `NULL` (default), will be randomized.
`sprobs`	the probability weights if using a randomized starting dose. If `NULL` (default) will be discrete-uniform.
`cohort`	the cohort (group) size, default 1.
`Fvals`	(vector or matrix): the true values of the response function on the dose grid. These are the dose-response scenarios from which the experimental runs will be simulated. If running an ensemble with different scenarios, each scenarios is a column. If running an identical-scenario ensemble, provide a single vector as well as `ensemble`.
`ensemble`	the number of different runs/scenarios to be simulated. Will be determined automatically if `Fvals` is a matrix, as the number of columns.
`design`	the dose-finding design function used to determine the next dose. Default `krow`; see `krow` for options.
`desArgs`	List of arguments passed on to `design`. Need to be compatible for use in `mapply`. Default is `list(k=1)`, which together with `design = krow` will generate a Classical (median-finding) UDD simulation.
`thresholds`	Matrix of size (at least) `n` by `ensemble`, the response thresholds of participants, presented as percentiles (i.e., output of `runif()`) rather than physical values. If `NULL` (default), they will be simulated on the fly. When running comparative performance simulations, we recommend providing the same thresholds to everything you want to compare on equal footing.
`seed`	The random seed if simulating the thresholds. Can be kept *"floating"* (i.e., varying between calls) if left as `NULL` (default).
`showdots`	Logical: print out a dot (`.`) after each designion step in `1:n`, and the start/end time stamps? Default `TRUE`.

Details

A vectorized dose-finding simulator, set up to run an entire ensemble simultaneously. The simulated doses are indices `1:nlev` with `nlev` being the number of dose levels. Upon output they can be optionally "dressed up" with physical values using the `xvals` argument.

The simulator's essential use within the `upndown` package is to estimate bootstrap confidence intervals for dose-averaging target estimates, via `dfboot`. But it can be also used stand-alone as a study-design aid.

The particular dose-finding design simulated is determined by `design` and its argument list `desArgs`. The 3 straightforward extensions of the median-finding "Classical" UDD are available, namely "k-in-a-row", biased-coin and group (cohort) UDD. To simulate the the median-finding "Classical" UDD itself, use `krow` with `desArgs = list(k=1)`. Other non-UDD dose-finding designs - e.g., CRM, CCD, BOIN, etc. - can also be made compatible with `dfsim`. Utilities to run those 3 in particular are available on GitHub, under `assaforon/UpndownBook/P3_Practical/OtherDesigns.r`.

If you want to create a `design` function yourself, it would need to accept `doses, responses` as input, and return the next dose allocation (as an integer index). The main progression loop is run via `mapply`.

Value

A list with the following elements:

- `scenarios`: `Fvals`
- `sample`: `thresholds`
- `doses`: The matrix of simulated dose allocations for each run (`n+1` by `ensemble`)
- `responses`: The matrix of simulated responses (0 or 1) for each run (`n` by `ensemble`)
- `cohort`: `cohort`
- `details`: `desArgs`

Note

This is an adaptation of a non-package function used by the author for well over a decade before incorporating it into upndown in late 2023. For initial guidance, see the code example. If you encounter any funny behavior please let me know. Thank you!

Author(s)

Assaf P. Oron

See Also

- `dfboot`

Examples

```
### dfsim example

# We provide a "toy example" for how randomized-F simulations might work.
# We have been strong advocated of this simulation approach, especially for performance comparison
#  between designs and between estimators. It is far preferable to the "cherry-picked F" approach.
# Unfortunately, the latter is still more commonly found in dose-finding literature.

# At core is a randomized ensemble of F(x) ("dose-response") curves. It is far easier
#   to generate parametric ensembles rather than nonparametric or semi-parametric ones.
# So this is what we do here. We use the Weibull, being capable of the most diverse ensembles
#   among common parametric families.

# This being a toy example, it is more simplistic than our current practice. For the latter,
#   see either the supplement to Oron et al. 2022, or the NE Journal of SDS due online
#   late 2024 or early 2025.  Ok, let's go!

# We simulate 7-level experiments.
m = 7
# And 10 curves in total
B = 10
# Parameter generation (I chose the parameter bounds after some trial and error)
# Note I am not fixing a seed, so each time you run this you'll get a different ensemble!
wparams = cbind(2^runif(B, -2, 2.5), runif(B, 1, 10) )
round(wparams, 2)
```

```
# Now the F(x) curves; they should be in columns
ensemble = apply(wparams, 1, function(x, doses = m)
                  pweibull(1:doses, shape = x[1], scale = x[2]) )

# Let's see what we got!
round(ensemble, 3)
# The experiment will be "Classical" median-targeting. In real life if 0.5 falls outside
#     of the range of doses, it's not great. For simulation it's fine; it'll enable us to
#     watch the allocations for such curves heap near the edge with F closest to 0.5.

# Let the experiments be a measly n=15, to keep runtime under the menacing 5 seconds.
# We start all runs smack in the middle, level 4:
sout = dfsim(n = 15, starting = 4, Fvals = ensemble, design = krow, desArgs = list(k=1) )
names(sout)

# "scenarios" are the F values we provided, while "sample" is the set of randomized thresholds
#   simulated from the uniform distribution. If you run comparisons, we recommend that
#   you only randomize the first set in the comparison, and feed the very same thresholds to
#   all the rest.

# Anyhoo, here are the simulated trajectories.
#     Note that there are n+1 x values in each one, because after
#     seeing x_n and y_n, the n+1-th allocation can be determined.
sout$doses
# Compare with the F ensemble. Is it what you expected?
# Probably more random-walky than you thought :)

# The binary responses are below. Before going big on the simulation, it's a good idea to
#   sanity-check and see that the doses and responses match according to the design's rules.
sout$responses
# Some meta details about the design and simulation settings are available here:
sout$details
```

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. `dynamean()` only uses `y` for bootstrap confidence intervals.
`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(x = doses, y = responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(x = doses, y = 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, conf = NULL)
# 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(x = doses, y = 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, conf = NULL)
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.
# This function now also provides bootstrap CIs, so we need to give it the y values.
# The default forces the final 2/3 of observations to be included; here in view of the long run-in
#     we are relaxing this
dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, conf = NULL)
# 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 `dynamean()` 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.
# Figure soon to follow.
# But first, have you noted how we keep specifying "conf = NULL"?
# This is because now at default, these averages calculate
# A bootstrap confidence-interval.
# These intervals are generally deficient but are the best anyone can do at present.

# If you want to use a confidence interval, you must provide the experiment's target,
#    or more precisely its balance point, as well as the parameters to use in
# the bootstrap simulation (which should be the ones generating the original experiment).

# Like this (not run, to avoid violating CRAN's very narrow limits on example runtime):
# dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, target = 0.5,
#                  design = krow, desArgs = list(k=1) )


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",
  allow1extra = FALSE,
  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, estsym` - 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 (CI). Default 90%. We do not recommend increasing to 95% unless you have ~ 100 or more observations. Setting to `NULL` if you don't want a CI. See more under "Value".
`estcol`, `estsize`, `estsymb`, `esthick`, `curvecol`	graphical parameters controlling the colors, symbol choice, size, thickness, of the target-dose and CIR-curve visuals.
`allow1extra`	logical: allow `length(x)` to be either equal or 1 greater than `length(y)`? (default `FALSE`) The *"n+1"* dose-allocation, determined from the last allocations and responses, might be tagged onto `x`. If this point is provided and `allow1extra=TRUE`, `udplot()` will show it as a grey diamond; the other functions will ignore it.
`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.

`dynamean` *Up-and-Down target-dose averaging estimate, with dynamic cutoff-point*

Description

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

Usage

```
dynamean(
  x,
  y = NULL,
  maxExclude = 1/2,
  before = FALSE,
  full = FALSE,
  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` or shorter by 1, and must be coded `TRUE/FALSE` or 0/1. `dynamean()` only uses `y` for bootstrap confidence intervals.
`maxExclude`	a fraction in $0, 1$ indicating the maximum initial fraction of the vector `x` to exclude from averaging, in case the algorithm-identified transition point occurs late in the experiment. Default 1/2.
`before`	logical: whether to start the averaging one observation earlier than the cutoff point. Default `FALSE`.
`full`	logical: should more detailed information be returned, or only the estimate? (default `FALSE`)
`conf`	the CI's confidence level, as a fraction in (0,1). To skip CI calculation set `conf = NULL`.
`...`	Additional parameters, mostly ones passed on to `dfboot` if `conf` is not `NULL`. See that function's help for details.

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 `dynamean()` 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).

Starting version 0.2.0, a bootstrap confidence interval (CI) is also provided. See `dfboot`, `dfsim` for additional parameters to pass to the bootstrap routine via `...`, beyond the confidence level `conf`. For the "Classical" median-finding UDD, use `design = krow, desArgs = list(k=1)`. To skip CI estimation, set `conf = NULL`. The experiment's binary responses (`y`) are only needed as input for confidence interval calculations; otherwise, only the dose-allocation sequence (`x`) is required.

`dynamean()` is recommended only for median-targeting UDDs, e.g., the "Classical" (traditional) design of Dixon and Mood. For off-median percentiles it might not work as well, and CI coverage will be lacking.

For such targets 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

If `full=FALSE` returns the point estimate. Otherwise returns a list with the index of the cutoff point used to start the averaging, and a 2-row matrix with full sequence of estimates and the tail vs. starting point indicator signs.

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(x = doses, y = responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(x = doses, y = 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, conf = NULL)
# 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(x = doses, y = 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, conf = NULL)
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.
# This function now also provides bootstrap CIs, so we need to give it the y values.
# The default forces the final 2/3 of observations to be included; here in view of the long run-in
#     we are relaxing this
dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, conf = NULL)
# 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 `dynamean()` 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.
# Figure soon to follow.
# But first, have you noted how we keep specifying "conf = NULL"?
# This is because now at default, these averages calculate
# A bootstrap confidence-interval.
# These intervals are generally deficient but are the best anyone can do at present.

# If you want to use a confidence interval, you must provide the experiment's target,
#    or more precisely its balance point, as well as the parameters to use in
# the bootstrap simulation (which should be the ones generating the original experiment).

# Like this (not run, to avoid violating CRAN's very narrow limits on example runtime):
# dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, target = 0.5,
#                  design = krow, desArgs = list(k=1) )


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 ;)
```

---

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, digits = 4)

ktargOptions(target, tolerance = 0.05, maxk = 20, digits = 4)

g2targ(cohort, lower, upper, digits = 4)

gtargOptions(target, minsize = 2, maxsize = 6, tolerance = 0.05, digits = 4)

bcoin(target, presentation = "both", digits = 4, tolerance = 0.05)
```

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?
`digits`	how many digits to round to? Default 4. Since the functions use R's `round()` utility, trailing zeroes will be truncated.
`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 0.05. • For `bcoin()`: the half-width of the interval around 0.5 in which the function recommends to simply use classical UD without a coin. Default 0.05.
`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).

`presentation`	`bcoin()` only: whether to report the coin probability as a `'decimal'`, rational `'fraction'`. or `'both'` (default).

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. The 0.05 default tolerance level around the target, is what we recommend as *"close enough"*. Otherwise, it is probably better to use biased-coin.

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 performance and operational simplicity. `bcoin()` can return the probability as a decimal, rational fraction, or both (default).

Value

- `k2targ(), g2targ()`: the official balance point given the user-provided design parameters.
- `ktargOptions(), gtargOptions()`: a `data.frame` with design parameters and balance points, 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.

`krow` *Up-and-Down Design Rules for use in Dose-Finding Simulator*

Description

Rules for k-in-a-row, Biased-Coin UD, and Group UD, coded as functions compatible with the generic dose-finding simulator `dfsim()`

These functions work on each virtual experimental run individually.

Usage

```
krow(doses, responses, k, lowTarget = NULL, fastStart = FALSE, ...)

bcd(doses, responses, coin, lowTarget, fastStart = FALSE, ...)

groupUD(doses, responses, s, ll, ul, ...)
```

Arguments

`doses, responses`	(mandatory arguments) vectors of the run's current sequence of doses (in ordinal/index scale) and responses
`k`	the number of consecutive identical responses required for dose transitions (k-in-a-row functions only).
`lowTarget`	(`krow` and `bcd`) logical: is the target below 0.5 (median threshold)?
`fastStart`	(`krow` and `bcd`) logical: should the experiment begin with a classical-UD-like stage until the first "minority" response is observed (i.e., a 1 for below-median targets and vice versa)? Even though `TRUE` delivers better experimental performance and is recommended when allowed, default is `FALSE` in order to ensure the user makes a conscious decision to allow it.

`...`	Technical pass-through argument, to allow for flexibility when constructing design-comparison simulation ensembles.
`coin`	(`bcd` only) the biased-coin probability. Note that unlike `bcdmat()`, here the function does not figure out automatically the coin probability and upper/lower target location from the provided target.
`s`	(`groupUD` only) the group/cohort size, analogous to `cohort` in `gudmat()`. We use a different name here because `cohort` is already used in `dfsim` that calls these utilities.
`ll, ul`	(`groupUD` only) how many positive responses are allowed for a move upward, and how many are required for a move downward, respectively. Analogous to `lower, upper` in `gudmat()`. For example `s=3, ll=0, ul=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

Rules for some popular or well-studied non-up-and-down

Value

the next dose allocation

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, marginalize = TRUE, ...)

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

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 `...`. For the k-in-a-row design, use `kmatMarg` for `pivec()` and `kmatFull` otherwise.
`...`	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.
`marginalize`	logical (for `currentvec, cumulvec` when `matfun = kmatFull`) should the returned vector be marginalized over dose levels (`TRUE`, default), or should the full set with internal states be returned? If desired, see the note to `kmatFull`'s help page for more details.
`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(proportions = 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` for `pivec`, and otherwise `kmatFull`.

At present, these functions are unable to incorporate in the calculations the impact of the recommended "fast start" stage for k-in-a-row and biased-coin designs. Such a stage begins with a classic UD run, until the first "minority" outcome is encountered (y=1 for below-median targets and vice versa). Generally such a fast start would make small-sample probability vectors approach the asymptotic distribution more quickly.

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, Flournoy N. Up-and-Down: The Most Popular, Most Reliable, and Most Overlooked Dose-Finding Design. *New Eng J Stat Data Science* 2024; 1-12.
- 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.
- `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,
  conf = 0.9,
  maxExclude = NULL,
  full = FALSE,
  weth66revs = TRUE,
  evenrevs = !all,
  ...
)

reversals(y, x = NULL, directional = TRUE, evenrevs = TRUE)
```

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. `dynamean()` only uses `y` for bootstrap confidence intervals.
`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 cutoff point onwards, should all values of `x` be used (`TRUE`, default), or only reversal points as in the Wetherill et al. approach? If set to `FALSE`, then the `before` flag also defaults to `FALSE` regardless of user choice.
`before`	logical: whether to start the averaging one observation earlier than the cutoff point. Default `FALSE`.
`conf`	the CI's confidence level, as a fraction in (0,1). To skip CI calculation set `conf = NULL`.
`maxExclude`	a fraction in $0, 1$ indicating the maximum initial fraction of the vector `x` to exclude from averaging, in case the algorithm-identified transition point occurs late in the experiment. Default 1/2.
`full`	logical: should more detailed information be returned, or only the estimate? (default `FALSE`)
`weth66revs`	(in `reversmean()`) logical: identical to `directional`. The argument name used in `reversmean()` stems from the article that switched to this definition of reversals when introducing reversal-averaging.
`evenrevs`	logical: should only an even number of reversals be used, meaning that if the total number is odd the last one is discarded? Default `TRUE` per common practice. However, when setting `all=TRUE` it makes sense to also set `evenrevs=FALSE`.
`...`	Additional parameters, mostly ones passed on to `dfboot` if `conf` is not `NULL`. See that function's help for details.
`directional`	(in `reversals()`) logical: should reversals be defined as change in direction (i.e., `x`; `TRUE` which is default), or in response (`y`)?

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 beginning from some cutoff 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 cutoff point (which should perhaps be called the *'cut-on'* 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.

Starting version 0.2.0, a bootstrap confidence interval (CI) is also provided. See `dfboot`, `dfsim` for additional parameters to pass to the bootstrap routine via `...`, beyond the confidence level `conf`. For the "Classical" median-finding UDD, use `design = krow, desArgs = list(k=1)`. To skip CI estimation, set `conf = NULL`. `reversmean()` is compatible mostly with median-targeting UDDs such as the "Classical" (traditional) design of Dixon and Mood. For general UDD target estimation, particularly off-median targeting designs, 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.
- `dynamean`, 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(x = doses, y = responses)

# Not at all! Let us overlay this one in red
abline(h = dixonmood(x = doses, y = 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, conf = NULL)
# 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(x = doses, y = 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, conf = NULL)
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.
# This function now also provides bootstrap CIs, so we need to give it the y values.
# The default forces the final 2/3 of observations to be included; here in view of the long run-in
#     we are relaxing this
dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, conf = NULL)
# 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 `dynamean()` 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.
# Figure soon to follow.
# But first, have you noted how we keep specifying "conf = NULL"?
# This is because now at default, these averages calculate
# A bootstrap confidence-interval.
# These intervals are generally deficient but are the best anyone can do at present.

# If you want to use a confidence interval, you must provide the experiment's target,
#    or more precisely its balance point, as well as the parameters to use in
# the bootstrap simulation (which should be the ones generating the original experiment).

# Like this (not run, to avoid violating CRAN's very narrow limits on example runtime):
# dynamean(c(doses, dosePlus1), responses, maxExclude = 0.5, target = 0.5,
#                 design = krow, desArgs = list(k=1) )


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,
  allow1extra = FALSE,
  curvedCI = NULL,
  ...
)
```

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 (CI). Default 90%. We do not recommend increasing to 95% unless you have ~ 100 or more observations. Setting to `NULL` if you don't want a CI. See more under "Value".
`allow1extra`	logical: allow `length(x)` to be either equal or 1 greater than `length(y)`? (default `FALSE`) The *"n+1"* dose-allocation, determined from the last allocations and responses, might be tagged onto `x`. If this point is provided and `allow1extra=TRUE`, `udplot()` will show it as a grey diamond; the other functions will ignore it.
`curvedCI`	logical: should confidence-interval boundaries rely upon an outwardly-curving interpolation (`TRUE`) or linear? If `NULL` (default), it will be `TRUE` for targets outside the 40th-60th percentile range.
`...`	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

Geneally, 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`).

However, if `conf = NULL` only the point estimate will be returned. This is for compatibility with bootstrap confidence intervals (which we use for dose-averaging estimators but not for CIR), and with UD ensemble simulation in general.

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.

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

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 19.9–23.5).
#' 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,
  allow1extra = FALSE,
  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))`).
`allow1extra`	logical: allow `length(x)` to be either equal or 1 greater than `length(y)`? (default `FALSE`) The *"n+1"* dose-allocation, determined from the last allocations and responses, might be tagged onto `x`. If this point is provided and `allow1extra=TRUE`, `udplot()` will show it as a grey diamond; the other functions will ignore it.
`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`>

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 Also

- plot.DRtrace, cir package.
- drplot for the up-and-down dose-response and estimate plotting.
- 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 19.9–23.5).
#' 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, flatOK = TRUE)

checkNatural(k, parname, toolarge = 1000)

checkDose(x, maxfrac = 0.9)

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)
flatOK	logical (checkCDF() only) if the CDF is completely flat, should function issue a warning (TRUE, default) - or stop with an error (FALSE)?
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 $0.9n$. Function also gives a warning if number exceeds $n/2$, since typically this suggests the effective sample size around target is too small.
`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

bcd (krow), 23
bcdmat, 2, *23*, *25*, *26*
bcoin (k2targ), 21

checkCDF (validUDinput), 38
checkDose (validUDinput), 38
checkNatural (validUDinput), 38
checkResponse (validUDinput), 38
checkTarget (validUDinput), 38
classicmat (bcdmat), 2
cumulvec, *4*
cumulvec (pivec), 24
currentvec, *4*
currentvec (pivec), 24

dfboot, 5, *8*, *9*, *18*, *29*
dfsim, *6*, *7*, 7, *18*, *29*
dixonmood, 10
drplot, 14, *37*
dynamean, *6*, 16, *29*

g2targ, *4*
g2targ (k2targ), 21
groupUD (krow), 23
gtargOptions, *4*
gtargOptions (k2targ), 21
gudmat (bcdmat), 2

k2targ, *4*, 21, *26*
kmatFull (bcdmat), 2
kmatMarg (bcdmat), 2
krow, *8*, 23
ktargOptions, *4*
ktargOptions (k2targ), 21

legend, *16*, *36*

par, *16*, *36*
pivec, *4*, *23*, 24
plot, *15*, *36*
plot.doseResponse, *16*
plot.DRtrace, *37*

quickInverse, *34*

reversals (reversmean), 27
reversmean, *11*, *18*, 27

udest, *11*, *18*, *29*, 32
udplot, *15*, *16*, 35

validUDinput, 38