Package ‘multinomineq’

February 20, 2024

Type Package

Title Bayesian Inference for Multinomial Models with Inequality Constraints

Version 0.2.6

Date 2024-02-19

Maintainer Daniel W. Heck <daniel.heck@uni-marburg.de>

Description Implements Gibbs sampling and Bayes factors for multinomial models with linear inequality constraints on the vector of probability parameters. As special cases, the model class includes models that predict a linear order of binomial probabilities (e.g., p[1] < p[2] < p[3] < .50) and mixture models assuming that the parameter vector p must be inside the convex hull of a finite number of predicted patterns (i.e., vertices). A formal definition of inequality-constrained multinomial models and the implemented computational methods is provided in: Heck, D.W., & Davis-Stober, C.P. (2019). Multinomial models with linear inequality constraints: Overview and improvements of computational methods for Bayesian inference. Journal of Mathematical Psychology, 91, 70-87. <doi:10.1016/j.jmp.2019.03.004>. Inequality-constrained multinomial models have applications in the area of judgment and decision making to fit and test random utility models (Regenwetter, M., Dana, J., & Davis-Stober, C.P. (2011). Transitivity of preferences. Psychological Review, 118, 42–56, <doi:10.1037/a0021150>) or to perform outcome-based strategy classification to select the decision strategy that provides the best account for a vector of observed choice frequencies (Heck, D.W., Hilbig, B.E., & Moshagen, M. (2017). From information processing to decisions: Formalizing and comparing probabilistic choice models. Cognitive Psychology, 96, 26–40. <doi:10.1016/j.cogpsych.2017.05.003>).

License GPL-3

URL https://github.com/danheck/multinomineq

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports Rcpp (>= 0.12.11), parallel, Rglpk, quadprog, coda, RcppXPtrUtils

Suggests knitr, rmarkdown, testthat, covr

LinkingTo Rcpp, RcppArmadillo, RcppProgress

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation yes

Author Daniel W. Heck [aut, cre] (<https://orcid.org/0000-0002-6302-9252>)

Repository CRAN

Date/Publication 2024-02-20 09:20:02 UTC

R topics documented:

`multinomineq-package` *multinomineq: Bayesian Inference for Inequality-Constrained Multinomial Models*

Description

Implements Gibbs sampling and Bayes factors for multinomial models with convex, linear-inequality constraints on the probability parameters. This includes models that predict a linear order of binomial probabilities (e.g., p1 < p2 < p3 < .50) and mixture models, which assume that the parameter vector p must be inside the convex hull of a finite number of vertices.

Details

A formal definition of inequality-constrained multinomial models and the implemented computational methods for Bayesian inference is provided in:

- Heck, D. W., & Davis-Stober, C. P. (2019). Multinomial models with linear inequality constraints: Overview and improvements of computational methods for Bayesian inference. Manuscript under revision. `https://arxiv.org/abs/1808.07140`

Inequality-constrained multinomial models have applications in multiple areas in psychology, judgement and decision making, and beyond:

- Testing choice axioms such as transitivity and random utility theory (Regenwetter et al., 2012, 2014). See `regenwetter2012`
- Testing deterministic axioms of measurement and choice (Karabatsos, 2005; Myung et al., 2005).
- Multiattribute decisions for probabilistic inferences involving strategies such as Take-the-best (TTB) vs. weighted additive (WADD; Bröder & Schiffer, 2003; Heck et al., 2017) See `heck2017` and `hilbig2014`
- Fitting and testing nonparametric item response theory models (Karabatsos & Sheu, 2004). See `karabatsos2004`
- Statistical inference for order-constrained contingency tables (Klugkist et al., 2007, 2010). See `bf_nonlinear`

- Testing stochastic dominance of response time distributions (Heathcote et al., 2010). See `stochdom_bf`
- Cognitive diagnostic assessment (Hoijtink et al., 2014).

Author(s)

Daniel W. Heck

References

Bröder, A., & Schiffer, S. (2003). Bayesian strategy assessment in multi-attribute decision making. Journal of Behavioral Decision Making, 16(3), 193-213. doi:10.1002/bdm.442

Bröder, A., & Schiffer, S. (2003). Take The Best versus simultaneous feature matching: Probabilistic inferences from memory and effects of reprensentation format. Journal of Experimental Psychology: General, 132, 277-293. doi:10.1037/00963445.132.2.277

Heck, D. W., Hilbig, B. E., & Moshagen, M. (2017). From information processing to decisions: Formalizing and comparing probabilistic choice models. Cognitive Psychology, 96, 26-40. doi:10.1016/j.cogpsych.2017.05.003

Hilbig, B. E., & Moshagen, M. (2014). Generalized outcome-based strategy classification: Comparing deterministic and probabilistic choice models. Psychonomic Bulletin & Review, 21(6), 1431-1443. doi:10.3758/s1342301406430

Regenwetter, M., & Davis-Stober, C. P. (2012). Behavioral variability of choices versus structural inconsistency of preferences. Psychological Review, 119(2), 408-416. doi:10.1037/a0027372

Regenwetter, M., Davis-Stober, C. P., Lim, S. H., Guo, Y., Popova, A., Zwilling, C., … Messner, W. (2014). QTest: Quantitative testing of theories of binary choice. Decision, 1(1), 2-34. doi:10.1037/dec0000007

Karabatsos, G. (2005). The exchangeable multinomial model as an approach to testing deterministic axioms of choice and measurement. Journal of Mathematical Psychology, 49(1), 51-69. doi:10.1016/j.jmp.2004.11.001

Myung, J. I., Karabatsos, G., & Iverson, G. J. (2005). A Bayesian approach to testing decision making axioms. *Journal of Mathematical Psychology, 49*, 205-225. doi:10.1016/j.jmp.2005.02.004

Karabatsos, G., & Sheu, C.-F. (2004). Order-constrained Bayes inference for dichotomous models of unidimensional nonparametric IRT. Applied Psychological Measurement, 28(2), 110-125. doi:10.1177/0146621603260678

Hoijtink, H. (2011). Informative Hypotheses: Theory and Practice for Behavioral and Social Scientists. Boca Raton, FL: Chapman & Hall/CRC.

Hoijtink, H., Béland, S., & Vermeulen, J. A. (2014). Cognitive diagnostic assessment via Bayesian evaluation of informative diagnostic hypotheses. Psychological Methods, 19(1), 21–38. doi:10.1037/a0034176

Klugkist, I., & Hoijtink, H. (2007). The Bayes factor for inequality and about equality constrained models. Computational Statistics & Data Analysis, 51(12), 6367-6379. doi:10.1016/j.csda.2007.01.024

Klugkist, I., Laudy, O., & Hoijtink, H. (2010). Bayesian evaluation of inequality and equality constrained hypotheses for contingency tables. Psychological Methods, 15(3), 281-299. doi:10.1037/a0020137

Heathcote, A., Brown, S., Wagenmakers, E. J., & Eidels, A. (2010). Distribution-free tests of stochastic dominance for small samples. Journal of Mathematical Psychology, 54(5), 454-463. doi:10.1016/j.jmp.2010.06.005

See Also

Useful links:

- https://github.com/danheck/multinomineq

Ab_drop_fixed — *Drop fixed columns in the Ab-Representation*

Description

Often inequalities refer to all probability parameters of a multinomial distribution. This function allows to transform the inequalities into the appropriate format `A * x <b` with respect to the free parameters only.

Usage

```
Ab_drop_fixed(A, b, options)
```

Arguments

`A`	a matrix defining the convex polytope via `A*x <= b`. The columns of `A` do not include the last choice option per item type and thus the number of columns must be equal to `sum(options-1)` (e.g., the column order of `A` for `k = c(a1,a2,a2, b1,b2)` is `c(a1,a2, b1)`).
`b`	a vector of the same length as the number of rows of `A`.
`options`	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.

Examples

```
# p1 < p2 < p3 < p4
A4 <- matrix(
  c(
    1, -1, 0, 0,
    0, 1, -1, 0,
    0, 0, 1, -1
  ),
  nrow = 3, byrow = TRUE
)
b4 <- c(0, 0, 0)

# drop the fixed column for: p4 = (1-p1-p2-p3)
Ab_drop_fixed(A4, b4, options = c(4))
```

Ab_max *Automatic Construction of Ab-Representation for Common Inequality Constraints*

Description

Constructs the matrix A and vector b of the Ab-representation A*x < b for common inequality constraints such as "the probability j is larger than all others (Ab_max)" or "the probabilities are ordered (Ab_monotonicity)").

Usage

```
Ab_max(
  which_max,
  options,
  exclude = c(),
  exclude_fixed = FALSE,
  drop_fixed = TRUE
)
```

Arguments

which_max	vector of indices refering to probabilities that are assumed to be larger than the remaining probabilities (e.g., which_max=c(1,2) means that p1>p3, p1>p4,..., p2>p3, ...). Note that the indices refer to *all* probabilities/categories (including one fixed probability within each multinomial distribution).
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.
exclude	vector of indices refering to probabilities that are excluded from the construction of the order constraints (including the fixed probabilities).
exclude_fixed	whether to exclude the fixed probabilities (i.e., the last probability within each multinomial) from the construction of the order constraints. For example, if options=c(2,2,3) then the probabilities/columns 2, 4, and 7 are dropped (which is equivalent to exclude=c(2,4,7)). This option is usually appropriate for binomial probabilities (i.e., if options = c(2,2,2,...)), e.g., when the interest is in the probability of correct responding across different item types.
drop_fixed	whether to drop columns of A containing the fixed probabilities (i.e., the last probability within each multinomial). *after* construction of the inequalities.

Value

a list with the matrix A and the vectors b and options

Examples

```
# Example 1: Multinomial with 5 categories
# Hypothesis: p1 is larger than p2,p3,p4,p5
Ab_max(which_max = 1, options = 5)

# Example 2: Four binomial probabilities
# Hypothesis: p1 is larger than p2,p3,p4
Ab_max(which_max = 1, options = c(2, 2, 2, 2), exclude_fixed = TRUE)
```

`Ab_multinom` *Get Constraints for Product-Multinomial Probabilities*

Description

Get or add inequality constraints (or vertices) to ensure that multinomial probabilities are positive and sum to one for all choice options within each item type.

Usage

```
Ab_multinom(options, A = NULL, b = NULL, nonneg = FALSE)
```

Arguments

`options`	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
`A`	a matrix defining the convex polytope via `A*x` <= `b`. The columns of `A` do not include the last choice option per item type and thus the number of columns must be equal to `sum(options-1)` (e.g., the column order of `A` for `k = c(a1,a2,a2,b1,b2)` is `c(a1,a2, b1)`).
`b`	a vector of the same length as the number of rows of `A`.
`nonneg`	whether to add constraints that probabilities must be nonnegative

Details

If `A` and `b` are provided, the constraints are added to these inequality constraints.

See Also

`add_fixed`

Examples

```
# three binary and two ternary choices:
options <- c(2, 2, 2, 3, 3)
Ab_multinom(options)
Ab_multinom(options, nonneg = TRUE)
```

Ab_sort *Sort Inequalities by Acceptance Rate*

Description

Uses samples from the prior/posterior to order the inequalities by the acceptance rate.

Usage

```
Ab_sort(A, b, k = 0, options, M = 1000, drop_irrelevant = TRUE)
```

Arguments

A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.
k	optional: number of observed frequencies (only for posterior sampling).
options	optional: number of options per item type/category system. Uniform sampling on [0,1] for each parameter is used if omitted.
M	number of samples.
drop_irrelevant	whether to drop irrelevant constraints for probabilities such as theta[1] >= 0, theta[1] <= 1, or sum(theta) <= 1.

Details

Those constraints that are rejected most often are placed at the first positions. This can help when computing the encompassing Bayes factor and counting how many samples satisfy the constraints (e.g., count_binom or bf_multinom). Essentially, it becomes more likely that the while-loop for testing whether the inequalities hold can stop earlier, thus making the computation faster.

The function could also be helpful to improve the efficiency of the stepwise sampling implemented in count_binom and count_multinom. First, one can use accept-reject sampling to test the first few, rejected inequalities. Next, one can use a Gibbs sampler to draw samples conditional on the first constraints.

Examples

```
### Binomial probabilities
b <- c(0, 0, .30, .70, 1)
A <- matrix(
  c(
    -1, 1, 0, # p1 >= p2
    0, 1, -1, # p2 <= p3
    1, 0, 0, # p1 <=.30
    0, 1, 0, # p2 <= .70
```

```
    0, 0, 1
  ), # p3 <= 1 (redundant)
  ncol = 3, byrow = 2
)
Ab_sort(A, b)


### Multinomial probabilities
# prior sampling:
Ab_sort(A, b, options = 4)
# posterior sampling:
Ab_sort(A, b, k = c(10, 3, 2, 14), options = 4)
```

---

bf_binom *Bayes Factor for Linear Inequality Constraints*

Description

Computes the Bayes factor for product-binomial/-multinomial models with linear order-constraints (specified via: `A*x <= b` or the convex hull `V`).

Usage

```
bf_binom(k, n, A, b, V, map, prior = c(1, 1), log = FALSE, ...)

bf_multinom(
  k,
  options,
  A,
  b,
  V,
  prior = rep(1, sum(options)),
  log = FALSE,
  ...
)
```

Arguments

`k`	vector of observed response frequencies.
`n`	the number of choices per item type. If `k=n=0`, Bayesian inference is relies on the prior distribution only.
`A`	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities `x` that fulfill the order constraints `A*x <= b`.
`b`	a vector of the same length as the number of rows of `A`.

V	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, A and b are ignored). Similar as for A, columns of V omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
map	optional: numeric vector of the same length as k with integers mapping the frequencies k to the free parameters/columns of A/V, thereby allowing for equality constraints (e.g., map=c(1,1,2,2)). Reversed probabilities 1-p are coded by negative integers. Guessing probabilities of .50 are encoded by zeros. The default assumes different parameters for each item type: map=1:ncol(A)
prior	a vector with two positive numbers defining the shape parameters of the beta prior distributions for each binomial rate parameter.
log	whether to return the log-Bayes factor instead of the Bayes factor
...	further arguments passed to count_binom or count_multinom (e.g., M, steps).
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.

Details

For more control, use count_binom to specifiy how many samples should be drawn from the prior and posterior, respectively. This is especially recommended if the same prior distribution (and thus the same prior probability/integral) is used for computing BFs for multiple data sets that differ only in the observed frequencies k and the sample size n. In this case, the prior probability/proportion of the parameter space in line with the inequality constraints can be computed once with high precision (or even analytically), and only the posterior probability/proportion needs to be estimated separately for each unique vector k.

Value

a matrix with two columns (Bayes factor and SE of approximation) and three rows:

- `bf_0u`: constrained vs. unconstrained (saturated) model
- `bf_u0`: unconstrained vs. constrained model
- `bf_00'`: constrained vs. complement of inequality-constrained model (e.g., pi>.2 becomes pi<=.2; this assumes identical equality constraints for both models)

References

Karabatsos, G. (2005). The exchangeable multinomial model as an approach to testing deterministic axioms of choice and measurement. Journal of Mathematical Psychology, 49(1), 51-69. doi:10.1016/j.jmp.2004.11.001

Regenwetter, M., Davis-Stober, C. P., Lim, S. H., Guo, Y., Popova, A., Zwilling, C., … Messner, W. (2014). QTest: Quantitative testing of theories of binary choice. Decision, 1(1), 2-34. doi:10.1037/dec0000007

See Also

count_binom and count_multinom for for more control on the number of prior/posterior samples and bf_nonlinear for nonlinear order constraints.

Examples

```
k <- c(0, 3, 2, 5, 3, 7)
n <- rep(10, 6)

# linear order constraints:
#             p1 <p2 <p3 <p4 <p5 <p6 <.50
A <- matrix(
  c(
    1, -1, 0, 0, 0, 0,
    0, 1, -1, 0, 0, 0,
    0, 0, 1, -1, 0, 0,
    0, 0, 0, 1, -1, 0,
    0, 0, 0, 0, 1, -1,
    0, 0, 0, 0, 0, 1
  ),
  ncol = 6, byrow = TRUE
)
b <- c(0, 0, 0, 0, 0, .50)

# Bayes factor: unconstrained vs. constrained
bf_binom(k, n, A, b, prior = c(1, 1), M = 10000)
bf_binom(k, n, A, b, prior = c(1, 1), M = 2000, steps = c(2, 4, 5))
bf_binom(k, n, A, b, prior = c(1, 1), M = 1000, cmin = 200)
```

---

bf_equality — *Bayes Factor with Inequality and (Approximate) Equality Constraints*

Description

To obtain the Bayes factor for the equality constraints `C*x = d`, a sequence of approximations `abs(C*x - d) < delta` is used.

Usage

```
bf_equality(
  k,
  options,
  A,
  b,
  C,
  d,
  prior = rep(1, sum(options)),
  M1 = 1e+05,
```

```
  M2 = 20000,
  delta = 0.5^(1:8),
  return_Ab = FALSE,
  ...
)
```

Arguments

k	the number of choices for each alternative ordered by item type (e.g. c(a1,a2,a3,b1,b2) for a ternary and a binary item type). The length of k must be equal to the sum of options. The default k=0 is equivalent to sampling from the prior.
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.
A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.
C	a matrix specifying the equality constraints C*x = d with columns refering to the free parameters (similar to A)
d	a vector with the same number of elements as the rows of C.
prior	the prior parameters of the Dirichlet-shape parameters. Must have the same length as k.
M1	number of independent samples from the encompassing model to test whether A*x < b.
M2	number of Gibbs-sampling iterations for each step of the approximation of C*x = d.
delta	a vector of stepsizes that are used for the approximation.
return_Ab	if TRUE, the function returns a list with the additional inequality constraints (specified via A, b, and steps) that are used in the stepwise approximation abs(C*x - d) < delta[i].
...	further arguments passed to count_binom or count_multinom (e.g., M, steps).

Details

First, the encompassing Bayes factor for the equality constraint A*x<b is computed using M1 independent Dirichlet samples. Next, the equality constraint C*x=d is approximated by drawing samples from the model A*x<b and testing whether abs(C*x - d) < delta[1]. Similarly, the stepsize delta is reduced step by step until abs(C*x - d) < min(delta). Klugkist et al. (2010) show that this procedure provides a valid approximation of the exact equality constraints if the step size becomes sufficiently small.

References

Klugkist, I., Laudy, O., & Hoijtink, H. (2010). Bayesian evaluation of inequality and equality constrained hypotheses for contingency tables. Psychological Methods, 15(3), 281-299. doi:10.1037/a0020137

Examples

```
# Equality constraints:  C * x = d
d <- c(.5, .5, 0)
C <- matrix(
  c(
    1, 0, 0, 0, # p1 = .50
    0, 1, 0, 0, # p2 = .50
    0, 0, 1, -1
  ), # p3 = p4
  ncol = 4, byrow = TRUE
)
k <- c(3, 7, 6, 4, 2, 8, 5, 5)
options <- c(2, 2, 2, 2)
bf_equality(k, options,
  C = C, d = d, delta = .5^(1:5),
  M1 = 50000, M2 = 5000
) # only for CRAN checks

# check against exact equality constraints (see ?bf_binom)
k_binom <- k[seq(1, 7, 2)]
bf_binom(k_binom,
  n = 10, A = matrix(0), b = 0,
  map = c(0, 0, 1, 1)
)
```

---

bf_nonlinear *Bayes Factor for Nonlinear Inequality Constraints*

Description

Computes the encompassing Bayes factor for a user-specified, nonlinear inequality constraint. Restrictions are defined via an indicator function of the free parameters `c(p11,p12,p13, p21,p22,...)` (i.e., the multinomial probabilities).

Usage

```
bf_nonlinear(
  k,
  options,
  inside,
  prior = rep(1, sum(options)),
  log = FALSE,
  ...
)

count_nonlinear(
  k = 0,
  options,
```

```
  inside,
  prior = rep(1, sum(options)),
  M = 5000,
  progress = TRUE,
  cpu = 1
)
```

Arguments

k	vector of observed response frequencies.
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
inside	an indicator function that takes a vector with probabilities `p=c(p11,p12, p21,p22,...)` (where the last probability for each multinomial is dropped) as input and returns `1` or `TRUE` if the order constraints are satisfied and `0` or `FALSE` otherwise (see details).
prior	a vector with two positive numbers defining the shape parameters of the beta prior distributions for each binomial rate parameter.
log	whether to return the log-Bayes factor instead of the Bayes factor
...	further arguments passed to `count_binom` or `count_multinom` (e.g., `M`, `steps`).
M	number of posterior samples drawn from the encompassing model
progress	whether a progress bar should be shown (if `cpu=1`).
cpu	either the number of CPUs used for parallel sampling, or a parallel cluster (e.g., `cl <- parallel::makeCluster(3)`). All arguments of the function call are passed directly to each core, and thus the total number of samples is `M*number_cpu`.

Details

Inequality constraints are defined via an indicator function `inside` which returns `inside(x)=1` (or `0`) if the vector of free parameters `x` is inside (or outside) the model space. Since the vector `x` must include only free (!) parameters, the last probability for each multinomial must not be used in the function `inside(x)`!

Efficiency can be improved greatly if the indicator function is defined as C++ code via the function cppXPtr in the package RcppXPtrUtils (see below for examples). In this case, please keep in mind that indexing in C++ starts with 0,1,2... (not with 1,2,3,... as in R)!

References

Klugkist, I., & Hoijtink, H. (2007). The Bayes factor for inequality and about equality constrained models. Computational Statistics & Data Analysis, 51(12), 6367-6379. doi:10.1016/j.csda.2007.01.024

Klugkist, I., Laudy, O., & Hoijtink, H. (2010). Bayesian evaluation of inequality and equality constrained hypotheses for contingency tables. Psychological Methods, 15(3), 281-299. doi:10.1037/a0020137

Examples

```
##### 2x2x2 continceny table (Klugkist & Hojtink, 2007)
#
# (defendant's race) x (victim's race) x (death penalty)
# indexing: 0 = white/white/yes  ; 1 = black/black/no
# probabilities: (p000,p001,  p010,p011,  p100,p101,  p110,p111)
# Model2:
# p000*p101 < p100*p001  &   p010*p111 < p110*p011

# observed frequencies:
k <- c(19, 132, 0, 9, 11, 52, 6, 97)

model <- function(x) {
  x[1] * x[6] < x[5] * x[2] & x[3] * (1 - sum(x)) < x[7] * x[4]
}
# NOTE: "1-sum(x)"  must be used instead of "x[8]"!

# compute Bayes factor (Klugkist 2007: bf_0u=1.62)
bf_nonlinear(k, 8, model, M = 50000)


##### Using a C++ indicator function (much faster)
cpp_code <- "SEXP model(NumericVector x){
  return wrap(x[0]*x[5] < x[4]*x[1] & x[2]*(1-sum(x)) < x[6]*x[3]);}"
# NOTE: C++ indexing starts at 0!

# define C++ pointer to indicator function:
model_cpp <- RcppXPtrUtils::cppXPtr(cpp_code)

bf_nonlinear(
  k = c(19, 132, 0, 9, 11, 52, 6, 97), M = 100000,
  options = 8, inside = model_cpp
)
```

---

binom_to_multinom *Converts Binary to Multinomial Frequencies*

Description

Converts the number of "hits" in the binary choice format to the observed frequencies across for all response categories (i.e., the multinomial format).

Usage

```
binom_to_multinom(k, n)
```

Arguments

- `k` vector of observed response frequencies.
- `n` the number of choices per item type. If `k=n=0`, Bayesian inference is relies on the prior distribution only.

Details

In `multinomineq`, binary choice frequencies are represented by the number of "hits" for each item type/condition (the vector `k`) and by the total number of responses per item type/condition (the scalar or vector `n`).

In the multinomial format, the vector `k` includes all response categories (not only the number of "hits"). This requires to define a vector `options`, which indicates how many categories belong to one item type/condition (since the total number of responses per item type is fixed).

Examples

```
k <- c(1, 5, 8, 10)
n <- 10
binom_to_multinom(k, n)
```

---

count_binom *Count How Many Samples Satisfy Linear Inequalities (Binomial)*

Description

Draws prior/posterior samples for product-binomial data and counts how many samples are inside the convex polytope defined by (1) the inequalities `A*x <= b` or (2) the convex hull over the vertices `V`.

Usage

```
count_binom(
  k,
  n,
  A,
  b,
  V,
  map,
  prior = c(1, 1),
  M = 10000,
  steps,
  start,
  cmin = 0,
  maxiter = 500,
  burnin = 5,
```

```
  progress = TRUE,
  cpu = 1
)
```

Arguments

k	vector of observed response frequencies.
n	the number of choices per item type. If k=n=0, Bayesian inference is relies on the prior distribution only.
A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.
V	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, A and b are ignored). Similar as for A, columns of V omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
map	optional: numeric vector of the same length as k with integers mapping the frequencies k to the free parameters/columns of A/V, thereby allowing for equality constraints (e.g., map=c(1,1,2,2)). Reversed probabilities 1-p are coded by negative integers. Guessing probabilities of .50 are encoded by zeros. The default assumes different parameters for each item type: map=1:ncol(A)
prior	a vector with two positive numbers defining the shape parameters of the beta prior distributions for each binomial rate parameter.
M	number of posterior samples drawn from the encompassing model
steps	an integer vector that indicates the row numbers at which the matrix A is split for a stepwise computation of the Bayes factor (see details). M can be a vector with the number of samples drawn in each step from the (partially) order-constrained models using Gibbs sampling. If cmin>0, samples are drawn for each step until count[i]>=cmin.
start	only relevant if steps is defined or cmin>0: a vector with starting values in the interior of the polytope. If missing, an approximate maximum-likelihood estimate is used.
cmin	if cmin>0: minimum number of counts per step in the automatic stepwise procedure. If steps is not defined, steps=c(1,2,3,4,...) by default.
maxiter	if cmin>0: maximum number of sampling runs in the automatic stepwise procedure.
burnin	number of burnin samples per step that are discarded. Since the maximum-likelihood estimate is used as a start value (which is updated for each step in the stepwise procedure in count_multinom), the burnin number can be smaller than in other MCMC applications.
progress	whether a progress bar should be shown (if cpu=1).

cpu — either the number of CPUs used for parallel sampling, or a parallel cluster (e.g., `cl <- parallel::makeCluster(3)`). All arguments of the function call are passed directly to each core, and thus the total number of samples is `M*number_cpu`.

Details

Counts the number of random samples drawn from beta distributions that satisfy the convex, linear-inequalitiy constraints. The function is useful to compute the encompassing Bayes factor for testing inequality-constrained models (see `bf_binom`; Hojtink, 2011).

The stepwise computation of the Bayes factor proceeds as follows: If the steps are defined as `steps=c(5,10)`, the BF is computed in three steps by comparing: (1) Unconstrained model vs. inequalities in `A[1:5,]`; (2) use posterior based on inequalities in `A[1:5,]` and check inequalities `A[6:10,]`; (3) sample from A[1:10,] and check inequalities in `A[11:nrow(A),]` (i.e., all inequalities).

Value

a matrix with the columns

- `count`: number of samples in polytope / that satisfy order constraints
- `M`: the total number of samples in each step
- `steps`: the `"steps"` used to sample from the polytope (i.e., the row numbers of `A` that were considered stepwise)

with the attributes:

- `proportion`: estimated probability that samples are in polytope
- `se`: standard error of probability estimate
- `const_map`: logarithm of the binomial constants that have to be considered due to equality constraints

References

Hoijtink, H. (2011). Informative Hypotheses: Theory and Practice for Behavioral and Social Scientists. Boca Raton, FL: Chapman & Hall/CRC.

Fukuda, K. (2004). Is there an efficient way of determining whether a given point q is in the convex hull of a given finite set S of points in Rd? Retrieved from https://www.cs.mcgill.ca/~fukuda/soft/polyfaq/node22.html

See Also

`bf_binom`, `count_multinom`

Examples

```
### a set of linear order constraints:
### x1 < x2 < .... < x6 < .50
A <- matrix(
  c(
```

```
    1, -1, 0, 0, 0, 0,
    0, 1, -1, 0, 0, 0,
    0, 0, 1, -1, 0, 0,
    0, 0, 0, 1, -1, 0,
    0, 0, 0, 0, 1, -1,
    0, 0, 0, 0, 0, 1
  ),
  ncol = 6, byrow = TRUE
)
b <- c(0, 0, 0, 0, 0, .50)

### check whether a specific vector is inside the polytope:
A %*% c(.05, .1, .12, .16, .19, .23) <= b


### observed frequencies and number of observations:
k <- c(0, 3, 2, 5, 3, 7)
n <- rep(10, 6)

### count prior samples and compare to analytical result
prior <- count_binom(0, 0, A, b, M = 5000, steps = 1:4)
prior # to get the proportion: attr(prior, "proportion")
(.50)^6 / factorial(6)

### count posterior samples + get Bayes factor
posterior <- count_binom(k, n, A, b, M = 2000, steps = 1:4)
count_to_bf(posterior, prior)

### automatic stepwise algorithm
prior <- count_binom(0, 0, A, b, M = 500, cmin = 200)
posterior <- count_binom(k, n, A, b, M = 500, cmin = 200)
count_to_bf(posterior, prior)
```

`count_multinom` *Count How Many Samples Satisfy Linear Inequalities (Multinomial)*

Description

Draws prior/posterior samples for product-multinomial data and counts how many samples are inside the convex polytope defined by (1) the inequalities `A*x <= b` or (2) the convex hull over the vertices V.

Usage

```
count_multinom(
  k = 0,
  options,
  A,
  b,
```

```
  V,
  prior = rep(1, sum(options)),
  M = 5000,
  steps,
  start,
  cmin = 0,
  maxiter = 500,
  burnin = 5,
  progress = TRUE,
  cpu = 1
)
```

Arguments

`k`	the number of choices for each alternative ordered by item type (e.g. `c(a1,a2,a3,b1,b2)` for a ternary and a binary item type). The length of `k` must be equal to the sum of `options`. The default `k=0` is equivalent to sampling from the prior.
`options`	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
`A`	a matrix defining the convex polytope via `A*x <= b`. The columns of `A` do not include the last choice option per item type and thus the number of columns must be equal to `sum(options-1)` (e.g., the column order of `A` for `k = c(a1,a2,a2,b1,b2)` is `c(a1,a2, b1)`).
`b`	a vector of the same length as the number of rows of `A`.
`V`	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, `A` and `b` are ignored). Similar as for `A`, columns of `V` omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
`prior`	the prior parameters of the Dirichlet-shape parameters. Must have the same length as `k`.
`M`	number of posterior samples drawn from the encompassing model
`steps`	an integer vector that indicates the row numbers at which the matrix `A` is split for a stepwise computation of the Bayes factor (see details). `M` can be a vector with the number of samples drawn in each step from the (partially) order-constrained models using Gibbs sampling. If `cmin>0`, samples are drawn for each step until `count[i]>=cmin`.
`start`	only relevant if `steps` is defined or `cmin>0`: a vector with starting values in the interior of the polytope. If missing, an approximate maximum-likelihood estimate is used.
`cmin`	if `cmin>0`: minimum number of counts per step in the automatic stepwise procedure. If `steps` is not defined, `steps=c(1,2,3,4,...)` by default.
`maxiter`	if `cmin>0`: maximum number of sampling runs in the automatic stepwise procedure.

burnin	number of burnin samples per step that are discarded. Since the maximum-likelihood estimate is used as a start value (which is updated for each step in the stepwise procedure in `count_multinom`), the `burnin` number can be smaller than in other MCMC applications.
progress	whether a progress bar should be shown (if `cpu=1`).
cpu	either the number of CPUs used for parallel sampling, or a parallel cluster (e.g., `cl <- parallel::makeCluster(3)`). All arguments of the function call are passed directly to each core, and thus the total number of samples is `M*number_cpu`.

Value

a list with the elements

a matrix with the columns

- `count`: number of samples in polytope / that satisfy order constraints
- `M`: the total number of samples in each step
- `steps`: the `"steps"` used to sample from the polytope (i.e., the row numbers of `A` that were considered stepwise)

with the attributes:

- `proportion`: estimated probability that samples are in polytope
- `se`: standard error of probability estimate

References

Hoijtink, H. (2011). Informative Hypotheses: Theory and Practice for Behavioral and Social Scientists. Boca Raton, FL: Chapman & Hall/CRC.

See Also

`bf_multinom`, `count_binom`

Examples

```
### frequencies:
#           (a1,a2,a3, b1,b2,b3,b4)
k <- c(1, 5, 9, 5, 3, 7, 8)
options <- c(3, 4)

### linear order constraints
# a1<a2<a3   AND   b2<b3<.50
# (note: a2<a3 <=> a2 < 1-a1-a2 <=> a1+2*a2 < 1)
# matrix A:
#             (a1,a2, b1,b2,b3)
A <- matrix(
  c(
    1, -1, 0, 0, 0,
    1, 2, 0, 0, 0,
    0, 0, 0, 1, -1,
```

```
    0, 0, 0, 0, 1
  ),
  ncol = sum(options - 1), byrow = TRUE
)
b <- c(0, 1, 0, .50)

# count prior and posterior samples and get BF
prior <- count_multinom(0, options, A, b, M = 2e4)
posterior <- count_multinom(k, options, A, b, M = 2e4)
count_to_bf(posterior, prior)

bf_multinom(k, options, A, b, M = 10000)
bf_multinom(k, options, A, b, cmin = 5000, M = 1000)
```

---

count_to_bf *Compute Bayes Factor Using Prior/Posterior Counts*

Description

Computes the encompassing Bayes factor (and standard error) defined as the ratio of posterior/prior samples that satisfy the order constraints (e.g., of a polytope).

Usage

```
count_to_bf(
  posterior,
  prior,
  exact_prior,
  log = FALSE,
  beta = c(1/2, 1/2),
  samples = 3000
)
```

Arguments

posterior	a vector (or matrix) with entries (or columns) `count` = number of posterior samples within polytope and `M` = total number of samples. See `count_binom`.
prior	a vecotr or matrix similar as for `posterior`, but based on samples from the prior distribution.
exact_prior	optional: the exact prior probabability of the order constraints. For instance, `exact_prior=1/factorial(4)` if pi1<pi2<pi3<pi4 (and if the prior is symmetric). If provided, `prior` is ignored.
log	whether to return the log-Bayes factor instead of the Bayes factor
beta	prior shape parameters of the beta distributions used for approximating the standard errors of the Bayes-factor estimates. The default is Jeffreys' prior.

`samples` number of samples from beta distributions used to compute standard errors.

The unconstrained (encompassing) model is the saturated baseline model that assumes a separate, independent probability for each observable frequency. The Bayes factor is obtained as the ratio of posterior/prior samples within an order-constrained subset of the parameter space.

The standard error of the (stepwise) encompassing Bayes factor is estimated by sampling ratios from beta distributions, with parameters defined by the posterior/prior counts (see Hoijtink, 2011; p. 211).

Value

a matrix with two columns (Bayes factor and SE of approximation) and three rows:

- `` `bf_0u` ``: constrained vs. unconstrained (saturated) model
- `` `bf_u0` ``: unconstrained vs. constrained model
- `` `bf_00'` ``: constrained vs. complement of inequality-constrained model (e.g., pi>.2 becomes pi<=.2; this assumes identical equality constraints for both models)

References

Hoijtink, H. (2011). Informative Hypotheses: Theory and Practice for Behavioral and Social Scientists. Boca Raton, FL: Chapman & Hall/CRC.

See Also

`count_binom`, `count_multinom`

Examples

```
# vector input
post <- c(count = 1447, M = 5000)
prior <- c(count = 152, M = 10000)
count_to_bf(post, prior)

# matrix input (due to nested stepwise procedure)
post <- cbind(count = c(139, 192), M = c(200, 1000))
count_to_bf(post, prior)

# exact prior probability known
count_to_bf(
  posterior = c(count = 1447, M = 10000),
  exact_prior = 1 / factorial(4)
)
```

`drop_fixed` *Drop or Add Fixed Dimensions for Multinomial Probabilities/Frequencies*

Description

Switches between two representation of polytopes for multinomial probabilities (whether the fixed parameters are included).

Usage

```
drop_fixed(x, options = 2)

add_fixed(x, options = 2, sum = 1)
```

Arguments

`x`	a vector (typically `k`, `n`, or `prior`) or a matrix (typically `A` or `V`), in which case the fixed dimensions are dropped/added column-wise.
`options`	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
`sum`	a vector that determines the fixed sum in each multinomial condition. By default, probabilities are assumed that sum to one. If frequencies `n` are provided, use `sum=n`.

Examples

```
######## bi- and trinomial (a1,a2, b1,b2,b3)
# vectors with frequencies:
drop_fixed(c(3, 7, 4, 1, 5), options = c(2, 3))
add_fixed(c(3, 4, 1),
  options = c(2, 3),
  sum = c(10, 10)
)

# matrices with probabilities:
V <- matrix(c(
  1, 0, 0,
  1, .5, .5,
  0, 1, 0
), 3, byrow = TRUE)
V2 <- add_fixed(V, options = c(2, 3))
V2
drop_fixed(V2, c(2, 3))
```

find_inside — *Find a Point/Parameter Vector Within a Convex Polytope*

Description

Finds the center/a random point that is within the convex polytope defined by the linear inequalities A*x <= b or by the convex hull over the vertices in the matrix V.

Usage

```
find_inside(
  A,
  b,
  V,
  options = NULL,
  random = FALSE,
  probs = TRUE,
  boundary = 1e-05
)
```

Arguments

A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.
V	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, A and b are ignored). Similar as for A, columns of V omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
options	optional: number of options per item type (only for $Ax \leq b$ representation). Necessary to account for sum-to-one constraints within multinomial distributions (e.g., p_1 + p_2 + p_3 <= 1). By default, parameters are assumed to be independent.
random	if TRUE, random starting values in the interior are generated. If FALSE, the center of the polytope is computed using linear programming.
probs	only for A*x<b representation: whether to add inequality constraints that the variables are probabilities (nonnegative and sum to 1 within each option)
boundary	constant value c that is subtracted on the right-hand side of the order constraints, $Ax \leq b - c$. This ensuresa that the resulting point is in the interior of the polytope and not at the boundary, which is important for MCMC sampling.

Details

If vertices V are provided, a convex combination of the vertices is returned. If random=TRUE, the weights are drawn uniformly from a Dirichlet distribution.

If inequalities are provided via A and b, linear programming (LP) is used to find the Chebyshev center of the polytope (i.e., the center of the largest ball that fits into the polytope; the solution may not be unique). If random=TRUE, LP is used to find a random point (not uniformly sampled!) in the convex polytope.

Examples

```
# inequality representation (A*x <= b)
A <- matrix(
  c(
    1, -1, 0, 1, 0,
    0, 0, -1, 0, 1,
    0, 0, 0, 1, -1,
    1, 1, 1, 1, 0,
    1, 1, 1, 0, 0,
    -1, 0, 0, 0, 0
  ),
  ncol = 5, byrow = TRUE
)
b <- c(0.5, 0, 0, .7, .4, -.2)
find_inside(A, b)
find_inside(A, b, random = TRUE)


# vertex representation
V <- matrix(c(
  # strict weak orders
  0, 1, 0, 1, 0, 1, # a < b < c
  1, 0, 0, 1, 0, 1, # b < a < c
  0, 1, 0, 1, 1, 0, # a < c < b
  0, 1, 1, 0, 1, 0, # c < a < b
  1, 0, 1, 0, 1, 0, # c < b < a
  1, 0, 1, 0, 0, 1, # b < c < a

  0, 0, 0, 1, 0, 1, # a ~ b < c
  0, 1, 0, 0, 1, 0, # a ~ c < b
  1, 0, 1, 0, 0, 0, # c ~ b < a
  0, 1, 0, 1, 0, 0, # a < b ~ c
  1, 0, 0, 0, 0, 1, # b < a ~ c
  0, 0, 1, 0, 1, 0, # c < a ~ b

  0, 0, 0, 0, 0, 0 # a ~ b ~ c
), byrow = TRUE, ncol = 6)
find_inside(V = V)
find_inside(V = V, random = TRUE)
```

heck2017 *Data: Multiattribute Decisions (Heck, Hilbig & Moshagen, 2017)*

Description

Choice frequencies with multiattribute decisions across 4 item types (Heck, Hilbig & Moshagen, 2017).

Usage

```
heck2017
```

Format

A data frame 4 variables:

- `B1` Frequency of choosing Option B for Item Type 1
- `B2` Frequency of choosing Option B for Item Type 2
- `B3` Frequency of choosing Option B for Item Type 3
- `B4` Frequency of choosing Option B for Item Type 4

Details

Each participant made 40 choices for each of 4 item types with four cues (with validities .9, .8, .7, and .6). The pattern of cue values of Option A and and B was as follows:

Item Type 1: A = (-1, 1, 1, -1) vs. B = (-1, -1, -1, -1)

Item Type 2: A = (1, -1, -1, 1) vs. B = (-1, 1, -1, 1)

Item Type 3: A = (-1, 1, 1, 1) vs. B = (-1, 1, 1, -1)

Item Type 4: A = (1, -1, -1, -1) vs. B = (-1, 1, 1, -1)

Raw data are available as `heck2017_raw`

References

Heck, D. W., Hilbig, B. E., & Moshagen, M. (2017). From information processing to decisions: Formalizing and comparing probabilistic choice models. Cognitive Psychology, 96, 26-40. doi:10.1016/j.cogpsych.2017.05.003

Examples

```
data(heck2017)
head(heck2017)
n <- rep(40, 4)

# cue validities and values
v <- c(.9, .8, .7, .6)
```

```
cueA <- matrix(
  c(
    -1, 1, 1, -1,
    1, -1, -1, 1,
    -1, 1, 1, 1,
    1, -1, -1, -1
  ),
  ncol = 4, byrow = TRUE
)
cueB <- matrix(
  c(
    -1, -1, -1, -1,
    -1, 1, -1, 1,
    -1, 1, 1, -1,
    -1, 1, 1, -1
  ),
  ncol = 4, byrow = TRUE
)

# get predictions
strategies <- c(
  "baseline", "WADDprob", "WADD",
  "TTBprob", "TTB", "EQW", "GUESS"
)
strats <- strategy_multiattribute(cueA, cueB, v, strategies)

# strategy classification with Bayes factor
strategy_postprob(heck2017[1:4, ], n, strats)
```

---

heck2017_raw *Data: Multiattribute Decisions (Heck, Hilbig & Moshagen, 2017)*

Description

Raw data with multiattribute decisions (Heck, Hilbig & Moshagen, 2017).

Usage

```
heck2017_raw
```

Format

A data frame with 21 variables:

- `vp` ID code of participant
- `trial` Trial index
- `pattern` Number of cue pattern
- `ttb` Prediction of take-the-best (TTB)
- `eqw` Prediction of equal weights (EQW)

`wadd` Prediction of weighted additive (WADD)

`logoddsdiff` Log-odds difference (WADDprob)

`ttbsteps` Number of TTB steps (TTBprob)

`itemtype` Item type as in paper

`reversedorder` Whether item is reversed

`choice` Choice

`rt` Response time

`choice.rev` Choice (reversed)

`a1` Value of Cue 1 for Option A

`a2` Value of Cue 2 for Option A

`a3` Value of Cue 3 for Option A

`a4` Value of Cue 4 for Option A

`b1` Value of Cue 1 for Option B

`b2` Value of Cue 2 for Option B

`b3` Value of Cue 3 for Option B

`b4` Value of Cue 4 for Option B

Details

Each participant made 40 choices for each of 4 item types with four cues (with validities .9, .8, .7, and .6). Individual choice freqeuncies are available as `heck2017`

References

Heck, D. W., Hilbig, B. E., & Moshagen, M. (2017). From information processing to decisions: Formalizing and comparing probabilistic choice models. Cognitive Psychology, 96, 26-40. doi:10.1016/j.cogpsych.2017.05.003

See Also

`heck2017` for the aggregated choice frequencies per item type.

Examples

```
data(heck2017_raw)
head(heck2017_raw)


# get cue values, validities, and predictions
cueA <- heck2017_raw[, paste0("a", 1:4)]
cueB <- heck2017_raw[, paste0("b", 1:4)]
v <- c(.9, .8, .7, .6)
strat <- strategy_multiattribute(
  cueA, cueB, v,
  c(
    "TTB", "TTBprob", "WADD",
```

```
    "WADDprob", "EQW", "GUESS"
  )
)

# get unique item types
types <- strategy_unique(strat)
types$unique

# get table of choice frequencies for analysis
freq <- with(
  heck2017_raw,
  table(vp, types$item_type, choice)
)
freqB <- freq[, 4:1, 1] + # reversed items: Option A
  freq[, 5:8, 2] # non-rev. items: Option B
head(40 - freqB)
data(heck2017)
head(heck2017) # same frequencies (different order)

# strategy classification
pp <- strategy_postprob(
  freqB[1:4, ], rep(40, 4),
  types$strategies
)
round(pp, 3)
```

`hilbig2014` *Data: Multiattribute Decisions (Hilbig & Moshagen, 2014)*

Description

Choice frequencies of multiattribute decisions across 3 item types (Hilbig & Moshagen, 2014).

Usage

```
hilbig2014
```

Format

A data frame 3 variables:

- `B1` Frequency of choosing Option B for Item Type 1
- `B2` Frequency of choosing Option B for Item Type 2
- `B3` Frequency of choosing Option B for Item Type 3

Details

Each participant made 32 choices for each of 3 item types with four cues (with validities .9, .8, .7, and .6).

The pattern of cue values of Option A and and B was as follows:

Item Type 1: A = (1, 1, 1, -1) vs. B = (-1, 1, -1, 1)

Item Type 2: A = (1, -1, -1, -1) vs. B = (-1, 1, 1, -1)

Item Type 3: A = (1, 1, 1, -1) vs. B = (-1, 1, 1, 1)

References

Hilbig, B. E., & Moshagen, M. (2014). Generalized outcome-based strategy classification: Comparing deterministic and probabilistic choice models. Psychonomic Bulletin & Review, 21(6), 1431-1443. doi:10.3758/s1342301406430

Examples

```
data(hilbig2014)
head(hilbig2014)

# validities and cue values
v <- c(.9, .8, .7, .6)
cueA <- matrix(
  c(
    1, 1, 1, -1,
    1, -1, -1, -1,
    1, 1, 1, -1
  ),
  ncol = 4, byrow = TRUE
)
cueB <- matrix(
  c(
    -1, 1, -1, 1,
    -1, 1, 1, -1,
    -1, 1, 1, 1
  ),
  ncol = 4, byrow = TRUE
)

# get strategy predictions
strategies <- c(
  "baseline", "WADDprob", "WADD",
  "TTB", "EQW", "GUESS"
)
preds <- strategy_multiattribute(cueA, cueB, v, strategies)
c <- c(1, rep(.5, 5)) # upper bound of probabilities

# use Bayes factor for strategy classification
n <- rep(32, 3)
strategy_postprob(k = hilbig2014[1:5, ], n, preds)
```

inside *Check Whether Points are Inside a Convex Polytope*

Description

Determines whether a point x is inside a convex poltyope by checking whether (1) all inequalities A*x <= b are satisfied or (2) the point x is in the convex hull of the vertices in V.

Usage

```
inside(x, A, b, V)
```

Arguments

x	a vector of length equal to the number of columns of A or V (i.e., a single point in D-dimensional space) or matrix of points/vertices (one per row).
A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.
V	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, A and b are ignored). Similar as for A, columns of V omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.

See Also

Ab_to_V and V_to_Ab to change between A/b and V representation.

Examples

```
# linear order constraints:  x1<x2<x3<.5
A <- matrix(c(
  1, -1, 0,
  0, 1, -1,
  0, 0, 1
), ncol = 3, byrow = TRUE)
b <- c(0, 0, .50)

# vertices: admissible points (corners of polytope)
V <- matrix(c(
  0, 0, 0,
  0, 0, .5,
  0, .5, .5,
  .5, .5, .5
), ncol = 3, byrow = TRUE)
```

```
xin <- c(.1, .2, .45) # inside
inside(xin, A, b)
inside(xin, V = V)

xout <- c(.4, .1, .55) # outside
inside(xout, A, b)
inside(xout, V = V)
```

---

inside_binom *Check Whether Choice Frequencies are in Polytope*

Description

Computes relative choice frequencies and checks whether these are in the polytope defined via (1) `A*x` <= `b` or (2) by the convex hull of a set of vertices `V`.

Usage

```
inside_binom(k, n, A, b, V)

inside_multinom(k, options, A, b, V)
```

Arguments

`k`	choice frequencies. For `inside_binom`: per item type (e.g.: a1,b1,c1,..) For `inside_multinom`: for all choice options ordered by item type (e.g., for ternary choices: a1,a2,a3, b1,b2,b3,..)
`n`	only for `inside_binom`: number of choices per item type.
`A`	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities `x` that fulfill the order constraints `A*x` <= `b`.
`b`	a vector of the same length as the number of rows of `A`.
`V`	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, `A` and `b` are ignored). Similar as for `A`, columns of `V` omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
`options`	only for `inside_multinom`: number of response options per item type.

See Also

`inside`

Examples

```
############ binomial
# x1<x2<x3<.50:
A <- matrix(c(
  1, -1, 0,
  0, 1, -1,
  0, 0, 1
), ncol = 3, byrow = TRUE)
b <- c(0, 0, .50)
k <- c(0, 1, 5)
n <- c(10, 10, 10)
inside_binom(k, n, A, b)

############ multinomial
# two ternary choices:
#     (a1,a2,a3,   b1,b2,b3)
k <- c(1, 4, 10, 5, 9, 1)
options <- c(3, 3)
# a1<b1, a2<b2, no constraints on a3, b3
A <- matrix(c(
  1, -1, 0, 0,
  0, 0, 1, -1
), ncol = 4, byrow = TRUE)
b <- c(0, 0)
inside_multinom(k, options, A, b)

# V-representation:
V <- matrix(c(
  0, 0, 0, 0,
  0, 0, 0, 1,
  0, 1, 0, 0,
  0, 0, 1, 1,
  0, 1, 0, 1,
  1, 1, 0, 0,
  0, 1, 1, 1,
  1, 1, 0, 1,
  1, 1, 1, 1
), 9, 4, byrow = TRUE)
inside_multinom(k, options, V = V)
```

---

karabatsos2004 *Data: Item Responses Theory (Karabatsos & Sheu, 2004)*

Description

The test was part of the 1992 Trial State Assessment in Reading at Grade 4, conducted by the National Assessments of Educational Progress (NAEP).

Usage

```
karabatsos2004
```

Format

A list with 4 matrices:

- `k.M`: Number of correct responses for participants with rest scores j=0,...,5 (i.e., the sum score minus the score for item i)
- `n.M`: Total number of participants for each cell of matrix `k.M`
- `k.IIO`: Number of correct responses for participants with sum scores j=0,...,6
- `n.IIO`: Total number of participants for each cell of matrix `k.IIO`

References

Karabatsos, G., & Sheu, C.-F. (2004). Order-constrained Bayes inference for dichotomous models of unidimensional nonparametric IRT. Applied Psychological Measurement, 28(2), 110-125. doi:10.1177/0146621603260678

See Also

The polytope for the nonparametric item response theory can be obtained using (see `nirt_to_Ab`).

Examples

```
data(karabatsos2004)
head(karabatsos2004)

######################################################
##### Testing Monotonicity (M)                   #####
##### (Karabatsos & Sheu, 2004, Table 3, p. 120) #####

IJ <- dim(karabatsos2004$k.M)
monotonicity <- nirt_to_Ab(IJ[1], IJ[2], axioms = "W1")
p <- sampling_binom(
  k = c(karabatsos2004$k.M),
  n = c(karabatsos2004$n.M),
  A = monotonicity$A, b = monotonicity$b,
  prior = c(.5, .5), M = 300
)

# posterior means (Table 4, p. 120)
post.mean <- matrix(apply(p, 2, mean), IJ[1],
  dimnames = dimnames(karabatsos2004$k.M)
)
round(post.mean, 2)

# posterior predictive checks (Table 4, p. 121)
ppp <- ppp_binom(p, c(karabatsos2004$k.M), c(karabatsos2004$n.M),
  by = 1:prod(IJ)
)
ppp <- matrix(ppp[, 3], IJ[1], dimnames = dimnames(karabatsos2004$k.M))
round(ppp, 2)
```

```
####################################################
##### Testing invariant item ordering (IIO)      #####
##### (Karabatsos & Sheu, 2004, Table 6, p. 122) #####

IJ <- dim(karabatsos2004$k.IIO)
iio <- nirt_to_Ab(IJ[1], IJ[2], axioms = "W2")
p <- sampling_binom(
  k = c(karabatsos2004$k.IIO),
  n = c(karabatsos2004$n.IIO),
  A = iio$A, b = iio$b,
  prior = c(.5, .5), M = 300
)
# posterior predictive checks (Table 6, p. 122)
ppp <- ppp_binom(prob = p, k = c(karabatsos2004$k.IIO),
                 n = c(karabatsos2004$n.IIO), by = 1:prod(IJ))
matrix(ppp[,3], 7, dimnames = dimnames(karabatsos2004$k.IIO))

# for each item:
ppp <- ppp_binom(p, c(karabatsos2004$k.IIO), c(karabatsos2004$n.IIO),
                 by = rep(1:IJ[2], each = IJ[1]))
round(ppp[,3], 2)
```

---

ml_binom *Maximum-likelihood Estimate*

Description

Get ML estimate for product-binomial/multinomial model with linear inequality constraints.

Usage

```
ml_binom(k, n, A, b, map, strategy, n.fit = 3, start, progress = FALSE, ...)

ml_multinom(k, options, A, b, V, n.fit = 3, start, progress = FALSE, ...)
```

Arguments

k	vector of observed response frequencies.
n	the number of choices per item type. If k=n=0, Bayesian inference is relies on the prior distribution only.
A	a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities x that fulfill the order constraints A*x <= b.
b	a vector of the same length as the number of rows of A.

map	optional: numeric vector of the same length as k with integers mapping the frequencies k to the free parameters/columns of A/V, thereby allowing for equality constraints (e.g., map=c(1,1,2,2)). Reversed probabilities 1-p are coded by negative integers. Guessing probabilities of .50 are encoded by zeros. The default assumes different parameters for each item type: map=1:ncol(A)
strategy	a list that defines the predictions of a strategy, seestrategy_multiattribute.
n.fit	number of calls to constrOptim.
start	only relevant if steps is defined or cmin>0: a vector with starting values in the interior of the polytope. If missing, an approximate maximum-likelihood estimate is used.
progress	whether a progress bar should be shown (if cpu=1).
...	further arguments passed to the function constrOptim. To ensure high accuracy, the number of maximum iterations should be sufficiently large (e.g., by setting control = list(maxit = 1e6, reltol=.Machine$double.eps^.6), outer.iterations = 1000.
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.
V	a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, A and b are ignored). Similar as for A, columns of V omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.

Details

First, it is checked whether the unconstrained maximum-likelihood estimator (e.g., for the binomial: k/n) is inside the constrained parameter space. Only if this is not the case, nonlinear optimization with convex linear-inequality constrained is used to estimate (A) the probability parameters θ for the Ab-representation or (B) the mixture weights α for the V-representation.

Value

the list returned by the optimizer constrOptim, including the input arguments (e.g., k, options, A, V, etc.).

- If the Ab-representation was used, par provides the ML estimate for the probability vector θ.
- If the V-representation was used, par provides the estimates for the (usually not identifiable) mixture weights α that define the convex hull of the vertices in V, while p provides the ML estimates for the probability parameters. Because the weights must sum to one, the α-parameter for the last row of the matrix V is dropped. If the unconstrained ML estimate is inside the convex hull, the mixture weights α are not estimated and replaced by missings (NA).

Examples

```
# predicted linear order: p1 < p2 < p3 < .50
# (cf. WADDprob in ?strategy_multiattribute)
```

```
A <- matrix(
  c(
    1, -1, 0,
    0, 1, -1,
    0, 0, 1
  ),
  ncol = 3, byrow = TRUE
)
b <- c(0, 0, .50)
ml_binom(k = c(4, 1, 23), n = 40, A, b)[1:2]
ml_multinom(
  k = c(4, 36, 1, 39, 23, 17),
  options = c(2, 2, 2), A, b
)[1:2]


# probabilistic strategy:  A,A,A,B  [e1<e2<e3<e4<.50]
strat <- list(
  pattern = c(-1, -2, -3, 4),
  c = .5, ordered = TRUE, prior = c(1, 1)
)
ml_binom(c(7, 3, 1, 19), 20, strategy = strat)[1:2]


# vertex representation (one prediction per row)
V <- matrix(c(
  # strict weak orders
  0, 1, 0, 1, 0, 1, # a < b < c
  1, 0, 0, 1, 0, 1, # b < a < c
  0, 1, 0, 1, 1, 0, # a < c < b
  0, 1, 1, 0, 1, 0, # c < a < b
  1, 0, 1, 0, 1, 0, # c < b < a
  1, 0, 1, 0, 0, 1, # b < c < a

  0, 0, 0, 1, 0, 1, # a ~ b < c
  0, 1, 0, 0, 1, 0, # a ~ c < b
  1, 0, 1, 0, 0, 0, # c ~ b < a
  0, 1, 0, 1, 0, 0, # a < b ~ c
  1, 0, 0, 0, 0, 1, # b < a ~ c
  0, 0, 1, 0, 1, 0, # c < a ~ b

  0, 0, 0, 0, 0, 0 # a ~ b ~ c
), byrow = TRUE, ncol = 6)
ml_multinom(
  k = c(4, 1, 5, 1, 9, 0, 7, 2, 1), n.fit = 1,
  options = c(3, 3, 3), V = V
)
```

`model_weights` *Get Posterior/NML Model Weights*

Description

Computes the posterior model probabilities based on the log-marginal likelihoods/negative NML values.

Usage

```
model_weights(x, prior)
```

Arguments

`x`	vector or matrix of log-marginal probabilities or negative NML values (if matrix: one model per column)
`prior`	vector of prior model probabilities (default: uniform over models). The vector is normalized internally to sum to one.

Examples

```
logmarginal <- c(-3.4, -2.0, -10.7)
model_weights(logmarginal)

nml <- matrix(c(
  2.5, 3.1, 4.2,
  1.4, 0.3, 8.2
), nrow = 2, byrow = TRUE)
model_weights(-nml)
```

`nirt_to_Ab` *Nonparametric Item Response Theory (NIRT)*

Description

Provides the inequality constraints on choice probabilities implied by nonparametric item response theory (NIRT; Karabatsos, 2001).

Usage

```
nirt_to_Ab(N, M, options = 2, axioms = c("W1", "W2"))
```

Arguments

`N`	number of persons / rows in item-response table
`M`	number of items / columns in item-response table
`options`	number of item categories/response options. If `options=2`, a dichotomous NIRT for product-binomial data is returned.
`axioms`	which axioms should be included in the polytope representation $A * x <= b$? See details.

Details

In contrast to parametric IRT models (e.g., the 1-parameter-logistic Rasch model), NIRT does not assume specific parametric shapes of the item-response and person-response functions. Instead, the necessary axioms for a unidimensional representation of the latent trait are tested directly.

The axioms are as follows:

"W1": Weak row/subject independence: Persons can be ordered on an ordinal scale independent of items.

"W2": Weak column/item independence: Items can be ordered on an ordinal scale independent of persons

"DC": Double cancellation: A necessary condition for a joint ordering of (person,item) pairs and an additive representation (i.e., an interval scale).

Note that axioms W1 and W2 jointly define the ISOP model by Scheiblechner (1995; isotonic ordinal probabilistic model) and the double homogeneity model by Mokken (1971). If DC is added, we obtain the ADISOP model by Scheiblechner (1999;).

References

Karabatsos, G. (2001). The Rasch model, additive conjoint measurement, and new models of probabilistic measurement theory. Journal of Applied Measurement, 2(4), 389–423.

Karabatsos, G., & Sheu, C.-F. (2004). Order-constrained Bayes inference for dichotomous models of unidimensional nonparametric IRT. Applied Psychological Measurement, 28(2), 110-125. doi:10.1177/0146621603260678

Mokken, R. J. (1971). A theory and procedure of scale analysis: With applications in political research (Vol. 1). Berlin: Walter de Gruyter.

Scheiblechner, H. (1995). Isotonic ordinal probabilistic models (ISOP). Psychometrika, 60(2), 281–304. doi:10.1007/BF02301417

Scheiblechner, H. (1999). Additive conjoint isotonic probabilistic models (ADISOP). Psychometrika, 64(3), 295–316. doi:10.1007/BF02294297

Examples

```
# 5 persons, 3 items
nirt_to_Ab(5, 3)
```

---

population_bf *Aggregation of Individual Bayes Factors*

Description

Aggregation of multiple individual (N=1) Bayes factors to obtain the evidence for a hypothesis in a population of persons.

Usage

```
population_bf(bfs)
```

Arguments

bfs	a vector with individual Bayes factors, a matrix with one type of Bayes-factor comparison per column, or a list of matrices with a named column "bf" (as returned by bf_multinom/count_to_bf).

Value

a vector or matrix with named elements/columns:

- population_bf: the product of individual BFs
- geometric_bf: the geometric mean of the individual BFs
- evidence_rate: the proportion of BFs>1 (BFs<1) if geometric_bf>1 (<1). Values close to 1.00 indicate homogeneity.
- stability_rate: the proportion bfs>geometric_bf (<) if geometric_bf>1 (<). Values close to 0.50 indicate stability.

References

Klaassen, F., Zedelius, C. M., Veling, H., Aarts, H., & Hoijtink, H. (in press). All for one or some for all? Evaluating informative hypotheses using multiple N = 1 studies. Behavior Research Methods. https://doi.org/10.3758/s13428-017-0992-5

Examples

```
# consistent evidence across persons:
bfs <- c(2.3, 1.8, 3.3, 2.8, 4.0, 1.9, 2.5)
population_bf(bfs)

# (A) heterogeneous, inconsistent evidence
# (B) heterogeneous, inconsistent evidence
bfs <- cbind(
  A = c(2.3, 1.8, 3.3, 2.8, 4.0, 1.9, 2.5),
  B = c(10.3, .7, 3.3, .8, 14.0, .9, 1.5)
)
population_bf(bfs)
```

postprob — *Transform Bayes Factors to Posterior Model Probabilities*

Description

Computes posterior model probabilities based on Bayes factors.

Usage

```
postprob(..., prior, include_unconstr = TRUE)
```

Arguments

`...` one or more Bayes-factor objects for different models as returned by the functions `bf_binom`, `bf_multinom` and `count_to_bf` (i.e., a 3x4 matrix containing a row "bf0u" and a column "bf"). Note that the Bayes factors must have been computed for the same data and using the same prior (this is not checked internally).

`prior` a vector of prior model probabilities (default: uniform). The order must be identical to that of the Bayes factors supplied via `...`. If `include_unconstr=TRUE`, the unconstrained model is automatically added to the list of models (at the last position).

`include_unconstr` whether to include the unconstrained, encompassing model without inequality constraints (i.e., the saturated model).

Examples

```
# data: binomial frequencies in 4 conditions
n <- 100
k <- c(59, 54, 74)

# Hypothesis 1: p1 < p2 < p3
A1 <- matrix(c(
  1, -1, 0,
  0, 1, -1
), 2, 3, TRUE)
b1 <- c(0, 0)

# Hypothesis 2: p1 < p2 and p1 < p3
A2 <- matrix(c(
  1, -1, 0,
  1, 0, -1
), 2, 3, TRUE)
b2 <- c(0, 0)

# get posterior probability for hypothesis
bf1 <- bf_binom(k, n, A = A1, b = b1)
```

```
bf2 <- bf_binom(k, n, A = A2, b = b2)
postprob(bf1, bf2,
  prior = c(bf1 = 1 / 3, bf2 = 1 / 3, unconstr = 1 / 3)
)
```

---

ppp_binom *Posterior Predictive p-Values*

Description

Uses posterior samples to get posterior-predicted frequencies and compare the Pearson's X^2 statistic for (1) the observed frequencies vs. (2) the posterior-predicted frequencies.

Usage

```
ppp_binom(prob, k, n, by)

ppp_multinom(prob, k, options, drop_fixed = TRUE)
```

Arguments

prob	vector with probabilities or a matrix with one probability vector per row. For rpbinom: probabilities of a success for each option. For rpmultinom: probabilities of all categories excluding the last category for each option (cf. drop_fixed). See also sampling_binom and sampling_multinom.
k	vector of observed response frequencies.
n	integer vector, specifying the number of trials for each binomial/multinomial distribution Note that this is the size argument in rmultinom, cf. Multinom.
by	optional: a vector of the same length as k indicating factor levels by which the posterior-predictive checks should be split (e.g., by item sets).
options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.
drop_fixed	whether the output matrix includes the last probability for each category (which is not a free parameter since probabilities must sum to one).

References

Myung, J. I., Karabatsos, G., & Iverson, G. J. (2005). A Bayesian approach to testing decision making axioms. *Journal of Mathematical Psychology, 49*, 205-225. doi:10.1016/j.jmp.2005.02.004

See Also

sampling_binom/sampling_multinom to get posterior samples and rpbinom/rpmultinom to get posterior-predictive samples.

Examples

```
# uniform samples:  p<.10
prob <- matrix(runif(300 * 3, 0, .1), 300)
n <- rep(10, 3)
ppp_binom(prob, c(1, 2, 0), n) # ok
ppp_binom(prob, c(5, 4, 3), n) # misfit

# multinomial (ternary choice)
prob <- matrix(runif(300 * 2, 0, .05), 300)
ppp_multinom(prob, c(1, 0, 9), 3) # ok
```

`regenwetter2012` *Data: Ternary Risky Choices (Regenwetter & Davis-Stober, 2012)*

Description

Raw data with choice frequencies for all 20 paired comparison of 5 gambles a, b, c, d, and e. Participants could either choose "Option 1", "Option 2", or "indifferent" (ternary choice). Each paired comparison (e.g., a vs. b) was repeated 45 times per participant. The data include 3 different gamble sets and aimed at testing whether people have transitive preferences (see Regenwetter & Davis-Stober, 2012).

Usage

```
regenwetter2012
```

Format

A matrix with 22 columns:

`participant`: Participant number

`gamble_set`: Gamble set

`a>b`: Number of times a preferred over b

`b>a`: Number of times b preferred over a

`a=b`: Number of times being indifferent between a and b

References

Regenwetter, M., & Davis-Stober, C. P. (2012). Behavioral variability of choices versus structural inconsistency of preferences. Psychological Review, 119(2), 408-416. doi:10.1037/a0027372

See Also

The substantive model of interest was the strict weak order polytope (see `swop5`).

Examples

```
data(regenwetter2012)
head(regenwetter2012)

# check transitive preferences: strict weak order polytope (SWOP)
data(swop5)
tail(swop5$A, 3)
# participant 1, gamble set 1:
p1 <- regenwetter2012[1, -c(1:2)]
inside_multinom(p1, swop5$options, swop5$A, swop5$b)


# posterior samples
p <- sampling_multinom(regenwetter2012[1, -c(1:2)],
  swop5$options, swop5$A, swop5$b,
  M = 100, start = swop5$start
)
colMeans(p)
apply(p[, 1:6], 2, plot, type = "l")
ppp_multinom(p, p1, swop5$options)

# Bayes factor
bf_multinom(regenwetter2012[1, -c(1:2)], swop5$options,
  swop5$A, swop5$b,
  M = 10000
)
```

---

rpbinom *Random Generation for Independent Multinomial Distributions*

Description

Generates random draws from independent multinomial distributions (= product-multinomial `pmultinom`).

Usage

```
rpbinom(prob, n)

rpmultinom(prob, n, options, drop_fixed = TRUE)
```

Arguments

`prob`	vector with probabilities or a matrix with one probability vector per row. For `rpbinom`: probabilities of a success for each option. For `rpmultinom`: probabilities of all categories excluding the last category for each option (cf. `drop_fixed`). See also `sampling_binom` and `sampling_multinom`.
`n`	integer vector, specifying the number of trials for each binomial/multinomial distribution Note that this is the `size` argument in `rmultinom`, cf. `Multinom`.

options	number of observable categories/probabilities for each item type/multinomial distribution, e.g., c(3,2) for a ternary and binary item.
drop_fixed	whether the output matrix includes the last probability for each category (which is not a free parameter since probabilities must sum to one).

Value

a matrix with one vector of frequencies per row. For rpbinom, only the frequencies of 'successes' are returned, whereas for rpmultinom, the complete frequency vectors (which sum to n within each option) are returned.

Examples

```
# 3 binomials
rpbinom(prob = c(.2, .7, .9), n = c(10, 50, 30))

# 2 and 3 options:  [a1,a2,  b1,b2,b3]
rpmultinom(
  prob = c(a1 = .5, b1 = .3, b2 = .6),
  n = c(10, 20), options = c(2, 3)
)
# or:
rpmultinom(
  prob = c(a1 = .5, a2 = .5, b1 = .3, b2 = .6, b3 = .1),
  n = c(10, 20), options = c(2, 3),
  drop_fixed = FALSE
)

# matrix with one probability vector per row:
p <- rpdirichlet(
  n = 6, alpha = c(1, 1, 1, 1, 1),
  options = c(2, 3)
)
rpmultinom(prob = p, n = c(20, 50), options = c(2, 3))
```

---

rpdirichlet *Random Samples from the Product-Dirichlet Distribution*

Description

Random samples from the prior/posterior (i.e., product-Dirichlet) of the unconstrained product-multinomial model (the encompassing model).

Usage

```
rpdirichlet(n, alpha, options, drop_fixed = TRUE)
```

Arguments

n	number of samples
alpha	Dirichlet parameters concatenated across independent conditions (e.g., a1,a2, b1,b2,b3)
options	the number of choice options per item type, e.g., `c(2,3)` for a binary and ternary condition. The sum of `options` must be equal to the length of `alpha`.
drop_fixed	whether the output matrix includes the last probability for each category (which is not a free parameter since probabilities must sum to one).

Examples

```
# standard uniform Dirichlet
rpdirichlet(5, c(1,1,1,1), 4)
rpdirichlet(5, c(1,1,1,1), 4, drop_fixed = FALSE)

# two ternary outcomes: (a1,a2,a3,  b1,b2,b3)
rpdirichlet(5, c(9,5,1,  3,6,6), c(3,3))
rpdirichlet(5, c(9,5,1,  3,6,6), c(3,3), drop_fixed = FALSE)
```

---

sampling_multinom *Posterior Sampling for Inequality-Constrained Multinomial Models*

Description

Uses Gibbs sampling to draw posterior samples for binomial and multinomial models with linear inequality-constraints.

Usage

```
sampling_multinom(
  k,
  options,
  A,
  b,
  V,
  prior = rep(1, sum(options)),
  M = 5000,
  start,
  burnin = 10,
  progress = TRUE,
  cpu = 1
)

sampling_binom(
  k,
  n,
```

```
  A,
  b,
  V,
  map = 1:ncol(A),
  prior = c(1, 1),
  M = 5000,
  start,
  burnin = 10,
  progress = TRUE,
  cpu = 1
)
```

Arguments

- `k` — the number of choices for each alternative ordered by item type (e.g. `c(a1,a2,a3,b1,b2)` for a ternary and a binary item type). The length of `k` must be equal to the sum of `options`. The default `k=0` is equivalent to sampling from the prior.
- `options` — number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
- `A` — a matrix with one row for each linear inequality constraint and one column for each of the free parameters. The parameter space is defined as all probabilities `x` that fulfill the order constraints `A*x <= b`.
- `b` — a vector of the same length as the number of rows of `A`.
- `V` — a matrix of vertices (one per row) that define the polytope of admissible parameters as the convex hull over these points (if provided, `A` and `b` are ignored). Similar as for `A`, columns of `V` omit the last value for each multinomial condition (e.g., a1,a2,a3,b1,b2 becomes a1,a2,b1). Note that this method is comparatively slow since it solves linear-programming problems to test whether a point is inside a polytope (Fukuda, 2004) or to run the Gibbs sampler.
- `prior` — the prior parameters of the Dirichlet-shape parameters. Must have the same length as `k`.
- `M` — number of posterior samples
- `start` — only relevant if `steps` is defined or `cmin>0`: a vector with starting values in the interior of the polytope. If missing, an approximate maximum-likelihood estimate is used.
- `burnin` — number of burnin samples that are discarded. Can be chosen to be small if the maxmimum-a-posteriori estimate is used as the (default) starting value.
- `progress` — whether a progress bar should be shown (if `cpu=1`).
- `cpu` — either the number of CPUs using separate MCMC chains in parallel, or a parallel cluster (e.g., `cl <- parallel::makeCluster(3)`). All arguments of the function call are passed directly to each core, and thus the total number of samples is `M*number_cpu`.
- `n` — the number of choices per item type. If `k=n=0`, Bayesian inference is relies on the prior distribution only.

map — optional: numeric vector of the same length as k with integers mapping the frequencies k to the free parameters/columns of A/V, thereby allowing for equality constraints (e.g., map=c(1,1,2,2)). Reversed probabilities 1-p are coded by negative integers. Guessing probabilities of .50 are encoded by zeros. The default assumes different parameters for each item type: map=1:ncol(A)

Details

Draws posterior samples for binomial/multinomial random utility models that assume a mixture over predefined preference orders/vertices that jointly define a convex polytope via the set of inequalities A * x < b or as the convex hull of a set of vertices V.

Value

an mcmc matrix (or an mcmc.list if cpu>1) with posterior samples of the binomial/multinomial probability parameters. See mcmc) .

References

Myung, J. I., Karabatsos, G., & Iverson, G. J. (2005). A Bayesian approach to testing decision making axioms. *Journal of Mathematical Psychology, 49*, 205-225. doi:10.1016/j.jmp.2005.02.004

See Also

count_multinom, ml_multinom

Examples

```
############### binomial ##########################
A <- matrix(
  c(
    1, 0, 0, # x1 < .50
    1, 1, 1, # x1+x2+x3 < 1
    0, 2, 2, # 2*x2+2*x3 < 1
    0, -1, 0, # x2 > .2
    0, 0, 1
  ), # x3 < .1
  ncol = 3, byrow = TRUE
)
b <- c(.5, 1, 1, -.2, .1)
samp <- sampling_binom(c(5, 12, 7), c(20, 20, 20), A, b)
head(samp)
plot(samp)


############### multinomial ##########################
# binary and ternary choice:
#           (a1,a2   b1,b2,b3)
k <- c(15, 9, 5, 2, 17)
options <- c(2, 3)

# columns:   (a1,  b1,b2)
```

```
A <- matrix(
  c(
    1, 0, 0, # a1 < .20
    0, 2, 1, # 2*b1+b2 < 1
    0, -1, 0, # b1 > .2
    0, 0, 1
  ), # b2 < .4
  ncol = 3, byrow = TRUE
)
b <- c(.2, 1, -.2, .4)
samp <- sampling_multinom(k, options, A, b)
head(samp)
plot(samp)
```

`sampling_nonlinear` *Posterior Sampling for Multinomial Models with Nonlinear Inequalities*

Description

A Gibbs sampler that draws posterior samples of probability parameters conditional on a (possibly nonlinear) indicator function defining a restricted parameter space that is convex.

Usage

```
sampling_nonlinear(
  k,
  options,
  inside,
  prior = rep(1, sum(options)),
  M = 1000,
  start,
  burnin = 10,
  eps = 1e-06,
  progress = TRUE,
  cpu = 1
)
```

Arguments

`k`	vector of observed response frequencies.
`options`	number of observable categories/probabilities for each item type/multinomial distribution, e.g., `c(3,2)` for a ternary and binary item.
`inside`	an indicator function that takes a vector with probabilities `p=c(p11,p12, p21,p22,...)` (where the last probability for each multinomial is dropped) as input and returns `1` or `TRUE` if the order constraints are satisfied and `0` or `FALSE` otherwise (see details).

`prior`	a vector with two positive numbers defining the shape parameters of the beta prior distributions for each binomial rate parameter.
`M`	number of posterior samples drawn from the encompassing model
`start`	only relevant if `steps` is defined or `cmin>0`: a vector with starting values in the interior of the polytope. If missing, an approximate maximum-likelihood estimate is used.
`burnin`	number of burnin samples that are discarded. Can be chosen to be small if the maxmimum-a-posteriori estimate is used as the (default) starting value.
`eps`	precision of the bisection algorithm
`progress`	whether a progress bar should be shown (if `cpu=1`).
`cpu`	either the number of CPUs used for parallel sampling, or a parallel cluster (e.g., `cl <- parallel::makeCluster(3)`). All arguments of the function call are passed directly to each core, and thus the total number of samples is `M*number_cpu`.

Details

Inequality constraints are defined via an indicator function `inside` which returns `inside(x)=1` (or `0`) if the vector of free parameters `x` is inside (or outside) the model space. Since the vector `x` must include only free (!) parameters, the last probability for each multinomial must not be used in the function `inside(x)`!

Efficiency can be improved greatly if the indicator function is defined as C++ code via the function cppXPtr in the package RcppXPtrUtils (see below for examples). In this case, please keep in mind that indexing in C++ starts with 0,1,2... (not with 1,2,3,... as in R)!

For each parameter, the Gibbs sampler draws a sample from the conditional posterior distribution (a scaled, truncated beta). The conditional truncation boundaries are computed with a bisection algorithm. This requires that the restricted parameteter space defined by the indicator function is convex.

Examples

```
# two binomial success probabilities: x = c(x1, x2)
# restriction to a circle:
model <- function(x) {
  (x[1] - .50)^2 + (x[2] - .50)^2 <= .15
}

# draw prior samples
mcmc <- sampling_nonlinear(
  k = 0, options = c(2, 2),
  inside = model, M = 1000
)
head(mcmc)
plot(c(mcmc[, 1]), c(mcmc[, 2]), xlim = 0:1, ylim = 0:1)


##### Using a C++ indicator function (much faster)
cpp_code <- "SEXP inside(NumericVector x){
  return wrap( sum(pow(x-.50, 2)) <= .15);}"
```

```
# NOTE: Uses Rcpp sugar syntax (vectorized sum & pow)

# define function via C++ pointer:
model_cpp <- RcppXPtrUtils::cppXPtr(cpp_code)
mcmc <- sampling_nonlinear(
  k = 0, options = c(2, 2),
  inside = model_cpp
)
head(mcmc)
plot(c(mcmc[, 1]), c(mcmc[, 2]), xlim = 0:1, ylim = 0:1)
```

`stochdom_Ab` *Ab-Representation for Stochastic Dominance of Histogram Bins*

Description

Provides the necessary linear equality constraints to test stochastic dominance of continuous distributions, that is, whether the cumulative density functions F satisfy the constraint $F_1(t) < F_2(t)$ for all t.

Usage

```
stochdom_Ab(bins, conditions = 2, order = "<")
```

Arguments

`bins`	number of bins of histogram
`conditions`	number of conditions
`order`	order constraint on the random variables across conditions. The default `order="<"` implies that the random variables increase across conditions (implying that the cdfs decrease: $F_1(t) > F_2(t)$).

References

Heathcote, A., Brown, S., Wagenmakers, E. J., & Eidels, A. (2010). Distribution-free tests of stochastic dominance for small samples. Journal of Mathematical Psychology, 54(5), 454-463. doi:10.1016/j.jmp.2010.06.005

See Also

`stochdom_bf` to obtain a Bayes factor directly.

Examples

```
stochdom_Ab(4, 2)
stochdom_Ab(4, 3)
```

stochdom_bf *Bayes Factor for Stochastic Dominance of Continuous Distributions*

Description

Uses discrete bins (as in a histogram) to compute the Bayes factor in favor of stochastic dominance of continuous distributions.

Usage

```
stochdom_bf(x1, x2, breaks = "Sturges", order = "<", ...)
```

Arguments

x1	a vector with samples from the first random variable/experimental condition.
x2	a vector with samples from the second random variable/experimental condition.
breaks	number of bins of histogram. See hist.
order	order constraint on the random variables across conditions. The default order="<" implies that the random variables increase across conditions (implying that the cdfs decrease: $F_1(t) > F_2(t)$).
...	further arguments passed to bf_multinom. Note that the noninformative default prior 1/number_of_bins is used.

References

Heathcote, A., Brown, S., Wagenmakers, E. J., & Eidels, A. (2010). Distribution-free tests of stochastic dominance for small samples. Journal of Mathematical Psychology, 54(5), 454-463. doi:10.1016/j.jmp.2010.06.005

Examples

```
x1 <- rnorm(300, 0, 1)
x2 <- rnorm(300, .5, 1) # dominates x1
x3 <- rnorm(300, 0, 1.2) # intersects x1

plot(ecdf(x1))
lines(ecdf(x2), col = "red")
lines(ecdf(x3), col = "blue")

b12 <- stochdom_bf(x1, x2, order = "<", M = 5e4)
b13 <- stochdom_bf(x1, x3, order = "<", M = 5e4)
b12$bf
b13$bf
```

strategy_marginal *Log-Marginal Likelihood for Decision Strategy*

Description

Computes the logarithm of the marginal likelihood, defined as the integral over the likelihood function weighted by the prior distribution of the error probabilities.

Usage

```
strategy_marginal(k, n, strategy)
```

Arguments

k	observed frequencies of Option B. Either a vector or a matrix/data frame (one person per row).
n	vector with the number of choices per item type.
strategy	a list that defines the predictions of a strategy, see`strategy_multiattribute`.

Examples

```
k <- c(1, 11, 18)
n <- c(20, 20, 20)
# pattern: A, A, B with constant error e<.50
strat <- list(
  pattern = c(-1, -1, 1),
  c = .5, ordered = FALSE,
  prior = c(1, 1)
)
m1 <- strategy_marginal(k, n, strat)
m1

# pattern: A, B, B with ordered error e1<e3<e2<.50
strat2 <- list(
  pattern = c(-1, 3, 2),
  c = .5, ordered = TRUE,
  prior = c(1, 1)
)
m2 <- strategy_marginal(k, n, strat2)
m2

# Bayes factor: Model 2 vs. Model 1
exp(m2 - m1)
```

strategy_multiattribute *Strategy Predictions for Multiattribute Decisions*

Description

Returns a list defining the predictions of different choice strategies (e.g., TTB, WADD)

Usage

```
strategy_multiattribute(cueA, cueB, v, strategy, c = 0.5, prior = c(1, 1))
```

Arguments

cueA	cue values of Option A (-1/+1 = negative/positive; 0 = missing). If a matrix is provided, each row defines one item type.
cueB	cue values of Option B (see cueA).
v	cue validities: probabilities that cues lead to correct decision. Must be of the same length as the number of cues.
strategy	strategy label, e.g., "TTB", "WADD", or "WADDprob". Can be a vector. See details.
c	defines the upper boundary for the error probabilities
prior	defines the prior distribution for the error probabilities (i.e., truncated independent beta distributions dbeta(prior[1], prior[2]))

Value

a strategy object (a list) with the entries:

- **pattern:** a numeric vector encoding the predicted choice pattern by the sign (negative = Option A, positive = Option B, 0 = guessing). Identical error probabilities are encoded by using the same absolute number (e.g., c(-1,1,1) defines one error probability with A,B,B predictions).
- **c:** upper boundary of error probabilities
- **ordered:** whether error probabilities are linearly ordered by their absolute value in pattern (largest error: smallest absolute number)
- **prior:** a numeric vector with two positive values specifying the shape parameters of the beta prior distribution (truncated to the interval [0,c]
- **label:** strategy label

Examples

```
# single item type
v <- c(.9, .8, .7, .6)
ca <- c(1, -1, -1, 1)
cb <- c(-1, 1, -1, -1)
strategy_multiattribute(ca, cb, v, "TTB")
```

```
strategy_multiattribute(ca, cb, v, "WADDprob")

# multiple item types
data(heck2017_raw)
strategy_multiattribute(
  heck2017_raw[1:10, c("a1", "a2", "a3", "a4")],
  heck2017_raw[1:10, c("b1", "b2", "b3", "b4")],
  v, "WADDprob"
)
```

---

strategy_postprob *Strategy Classification: Posterior Model Probabilities*

Description

Posterior model probabilities for multiple strategies (with equal prior model probabilities).

Usage

```
strategy_postprob(k, n, strategies, cpu = 1)
```

Arguments

k	observed frequencies of Option B. Either a vector or a matrix/data frame (one person per row).
n	vector with the number of choices per item type.
strategies	list of strategies. See strategy_multiattribute
cpu	number of processing units for parallel computation.

See Also

strategy_marginal and model_weights

Examples

```
# pattern 1: A, A, B with constant error e<.50
strat1 <- list(
  pattern = c(-1, -1, 1),
  c = .5, ordered = FALSE,
  prior = c(1, 1)
)
# pattern 2: A, B, B with ordered error e1<e3<e2<.50
strat2 <- list(
  pattern = c(-1, 3, 2),
  c = .5, ordered = TRUE,
  prior = c(1, 1)
)
baseline <- list(
```

```
  pattern = 1:3, c = 1, ordered = FALSE,
  prior = c(1, 1)
)

# data
k <- c(3, 4, 12) # frequencies Option B
n <- c(20, 20, 20) # number of choices
strategy_postprob(k, n, list(strat1, strat2, baseline))
```

---

strategy_to_Ab *Transform Pattern of Predictions to Polytope*

Description

Transforms ordered item-type predictions to polytope definition. This allows to use Monte-Carlo methods to compute the Bayes factor if the number of item types is large (bf_binom).

Usage

```
strategy_to_Ab(strategy)
```

Arguments

strategy	a decision strategy returned by strategy_multiattribute.

Details

Note: Only works for models without guessing predictions and without equality constraints (i.e., requires separate error probabilities per item type)

Value

a list containing the matrix A and the vector b that define a polytope via A*x <= b.

Examples

```
# strategy:  A,B,B,A   e2<e3<e4<e1<.50
strat <- list(
  pattern = c(-1, 4, 3, -2),
  c = .5, ordered = TRUE,
  prior = c(1, 1)
)
pt <- strategy_to_Ab(strat)
pt

# compare results to encompassing BF method:
b <- list(
  pattern = 1:4, c = 1,
  ordered = FALSE, prior = c(1, 1)
)
```

```
k <- c(2, 20, 18, 0)
n <- rep(20, 4)
m1 <- strategy_postprob(k, n, list(strat, b))
log(m1[1] / m1[2])
bf_binom(k, n, pt$A, pt$b, log = TRUE)
```

---

strategy_unique *Unique Patterns/Item Types of Strategy Predictions*

Description

Find unique item types, which are defined as patterns of cue values that lead to identical strategy predictions.

Usage

```
strategy_unique(strategies, add_baseline = TRUE, reversed = FALSE)
```

Arguments

strategies	a list of strategy predictions with the same length of the vector `pattern`, see strategy_multiattribute.
add_baseline	whether to add a baseline model which assumes one probability in [0,1] for each item type.
reversed	whether reversed patterns are treated separately (default: automatically switch Option A and B if `pattern=c(-1,1,1,1)`)

Value

a list including:

- `unique`: a matrix with the unique strategy patterns
- `item_type`: a vector that maps the original predictions to item types (negative: reversed items)
- `strategies`: a list with strategy predictions with `pattern` adapted to the unique item types

Examples

```
data(heck2017_raw)
ca <- heck2017_raw[1:100, c("a1", "a2", "a3", "a4")]
cb <- heck2017_raw[1:100, c("b1", "b2", "b3", "b4")]
v <- c(.9, .8, .7, .6)
strats <- strategy_multiattribute(
  ca, cb, v,
  c("WADDprob", "WADD", "TTB")
)
strategy_unique(strats)
```

swop5 *Strict Weak Order Polytope for 5 Elements and Ternary Choices*

Description

Facet-defining inequalities of the strict weak order mixture model for all 10 paired comparisons of 5 choice elements {a,b,c,d,e} in a 3-alternative forced-choice task (Regenwetter & Davis-Stober, 2012).

Usage

```
swop5
```

Format

A list with 3 elements:

- **A:** Matrix with inequality constraints that define a polytope via `A*x <= b`.
- **b:** vector with upper bounds for the inequalities.
- **start:** A point in the polytope.
- **options:** A vector with the number of options (=3) per item type.

References

Regenwetter, M., & Davis-Stober, C. P. (2012). Behavioral variability of choices versus structural inconsistency of preferences. Psychological Review, 119(2), 408-416. doi:10.1037/a0027372

See Also

The corresponding data set `regenwetter2012`.

Examples

```
data(swop5)
tail(swop5$A) # A*x <= b
tail(swop5$b)
swop5$start # inside SWOP polytope
swop5$options # 3 choice options per item

# check whether point is in polytope:
inside(swop5$start, swop5$A, swop5$b)


# get prior samples:
p <- sampling_multinom(0, swop5$options,
  swop5$A, swop5$b,
  M = 100, start = swop5$start
)
```

```
colMeans(p)
apply(p[, 1:5], 2, plot, type = "l")
```

V_to_Ab *Transform Vertex/Inequality Representation of Polytope*

Description

For convex polytopes: Requires rPorta (https://github.com/TasCL/rPorta) to transform the vertex representation to/from the inequality representation. Since rPorta cannot be compiled with R versions >=4.0.0 anymore, the function is currently deprecated.

Usage

```
V_to_Ab(V)

Ab_to_V(A, b, options = 2)
```

Arguments

V	a matrix with one vertex of a polytope per row (e.g., the admissible preference orders of a random utility model or any other theory). Since the values have to sum up to one within each multinomial condition, the last value of each multinomial is omitted (e.g., the prediction 1-0-0/0-1 for a tri and binomial becomes 1-0/0).
A	a matrix defining the convex polytope via A*x <= b. The columns of A do not include the last choice option per item type and thus the number of columns must be equal to sum(options-1) (e.g., the column order of A for k = c(a1,a2,a2, b1,b2) is c(a1,a2, b1)).
b	a vector of the same length as the number of rows of A.
options	number of choice options per item type. Can be a vector options=c(2,3,4) if item types have 2/3/4 choice options.

Details

Choice models can be represented as polytopes if they assume a latent mixture over a finite number preference patterns (random preference model). For the general approach and theory underlying binary and ternary choice models, see Regenwetter et al. (2012, 2014, 2017).

The function is currently deprecated since the package rPorta cannot be compiled with R>=4.0.0!

For binary choices (options=2), additional constraints are added to A and b to ensure that all dimensions of the polytope satisfy: 0 <= p_i <= 1. For ternary choices (options=3), constraints are added to ensure that 0 <= p_1+p_2 <=1 for pairwise columns (1+2, 3+4, 5+6, ...). See Ab_multinom.

References

Regenwetter, M., & Davis-Stober, C. P. (2012). Behavioral variability of choices versus structural inconsistency of preferences. Psychological Review, 119(2), 408-416. doi:10.1037/a0027372

Regenwetter, M., Davis-Stober, C. P., Lim, S. H., Guo, Y., Popova, A., Zwilling, C., … Messner, W. (2014). QTest: Quantitative testing of theories of binary choice. Decision, 1(1), 2-34. doi:10.1037/dec0000007

Regenwetter, M., & Robinson, M. M. (2017). The construct–behavior gap in behavioral decision research: A challenge beyond replicability. Psychological Review, 124(5), 533-550. https://doi.org/10.1037/rev0000067

Index