Package ‘xactonomial’

April 28, 2025

Type Package

Title Inference for Functions of Multinomial Parameters

Version 1.0.3

Date 2025-04-20

URL https://sachsmc.github.io/xactonomial/

BugReports https://github.com/sachsmc/xactonomial/issues/

Description We consider the problem where we observe k vectors (possibly of different lengths), each representing an independent multinomial random vector. For a given function that takes in the concatenated vector of multinomial probabilities and outputs a real number, this is a Monte Carlo estimation procedure of an exact p-value and confidence interval. The resulting inference is valid even in small samples, when the parameter is on the boundary, and when the function is not differentiable at the parameter value, all situations where asymptotic methods and the bootstrap would fail. For more details see Sachs, Fay, and Gabriel (2025) <doi:10.48550/arXiv.2406.19141>.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements Cargo (Rust's package manager), rustc >= 1.70

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.3.2

Config/rextendr/version 0.3.1.9001

Config/testthat/edition 3

Depends R (>= 4.2)

NeedsCompilation yes

Author Michael C Sachs [aut, cre],
Michael P Fay [aut],
Erin E Gabriel [aut],
David B Dahl [ctb] ((rbindings.rs))

Maintainer Michael C Sachs <sachsmc@gmail.com>

Repository CRAN

Date/Publication 2025-04-28 13:50:02 UTC

Contents

`calc_multinom_probs` *Calculate multinomial probabilities*

Description

Calculate multinomial probabilities

Usage

```
calc_multinom_probs(sar, logt, logc, d, n, nt)
```

Arguments

`sar`	The unrolled matrix containing the portion of the sample space to sum over
`logt`	The vector of candidate theta values, as sampled from the null space
`logc`	The vector of log multinomial coefficients see log_multinom_coef
`d`	The total dimension, sum(d_j)
`n`	The sample size
`nt`	The number of candidate theta values

Value

A vector of probabilities

Examples

```
sspace_3_5 <- sspace_multinom(3, 5)
calc_multinom_probs(sspace_3_5, sample_unit_simplexn(3, 10),
  apply(matrix(sspace_3_5, ncol = 3, byrow = TRUE), 1, log_multinom_coef, sumx = 5), 3, 5, 10)
```

---

calc_prob_null — *Calculate probability for given parameters*

Description

Given a set of candidate parameter vectors, the enumerated sample space, and a logical vector with the same number of elements of the sample space, compute the probability for each element of the sample space and take the sum.

Usage

```
calc_prob_null(theta_cands, SSpacearr, logC, II)
```

Arguments

`theta_cands`	A matrix with samples in the rows and the parameters in the columns
`SSpacearr`	A matrix with the sample space for the given size of the problem
`logC`	log multinomial coefficient for each element of the sample space
`II`	logical vector of sample space psi being more extreme than the observed psi

Value

A numeric vector of probabilities

Examples

```
sspace_3_5 <- matrix(sspace_multinom(3, 5), ncol = 3, byrow = TRUE)
theta_cands <- matrix(sample_unit_simplexn(3, 10), ncol = 3,byrow = TRUE)
calc_prob_null_fast(theta_cands, sspace_3_5,
apply(sspace_3_5, 1, log_multinom_coef, sumx = 5), II = 1:21 > 12)
# same as below but faster
calc_prob_null(theta_cands, sspace_3_5,
apply(sspace_3_5, 1, log_multinom_coef, sumx = 5), II = 1:21 > 12)
```

calc_prob_null_fast *Calculate probability for given parameters*

Description

Given a set of candidate parameter vectors, the enumerated sample space, and a logical vector with the same number of elements of the sample space, compute the probability for each element of the sample space and take the sum.

Usage

```
calc_prob_null_fast(theta_cands, SSpacearr, logC, II)
```

Arguments

theta_cands	A matrix with samples in the rows and the parameters in the columns
SSpacearr	A matrix with the sample space for the given size of the problem
logC	log multinomial coefficient for each element of the sample space
II	logical vector of sample space psi being more extreme than the observed psi

Value

A numeric vector of probabilities

Examples

```
sspace_3_5 <- matrix(sspace_multinom(3, 5), ncol = 3, byrow = TRUE)
theta_cands <- matrix(sample_unit_simplexn(3, 10), ncol = 3,byrow = TRUE)
calc_prob_null_fast(theta_cands, sspace_3_5,
apply(sspace_3_5, 1, log_multinom_coef, sumx = 5), II = 1:21 > 12)
# same as below but faster
calc_prob_null(theta_cands, sspace_3_5,
apply(sspace_3_5, 1, log_multinom_coef, sumx = 5), II = 1:21 > 12)
```

calc_prob_null_gradient *Gradient of the multinomial likelihood sum*

Description

Gradient of the multinomial likelihood sum

Usage

```
calc_prob_null_gradient(theta_cands, SSpacearr, II)
```

Arguments

`theta_cands`	A matrix with samples in the rows and the parameters in the columns
`SSpacearr`	A matrix with the sample space for the given size of the problem
`II`	logical vector of sample space psi being more extreme than the observed psi

Value

A matrix the same dimension as theta_cands

Examples

```
calc_prob_null_gradient(t(c(.28, .32, .4)),
matrix(c(2, 2, 1, 1, 2, 2, 0, 3, 2), ncol = 3),
rep(TRUE, 3))

# numerically
testenv <- new.env()
testenv$SSpacearr <- matrix(c(2, 2, 1, 1, 2, 2, 0, 3, 2), ncol = 3)
testenv$thistheta <- c(.28, .32, .4)
numericDeriv(quote(sum(exp((.colSums(t(SSpacearr) * log(thistheta), m = 3, n = 3))))),
    theta = "thistheta", rho = testenv, central = TRUE)
```

`combinate` *Arrange all combinations of rows of two matrices*

Description

Given X and Y, both matrices where the rows are counts of multinomial trials, produce all combinations rowwise, concatenate the rows into a new matrix, and calculate the log multinomial coefficients for the combination.

Usage

```
combinate(X, Y)
```

Arguments

`X`	Matrix 1
`Y`	Matrix 2

Value

A list containing Sspace, the sample space (vectors of counts), and logC, a vector of the log multinomial coefficients.

Examples

```
slist_2_3 <- combinate(matrix(sspace_multinom(2, 5), ncol = 2, byrow = TRUE),
   matrix(sspace_multinom(3, 6), ncol = 3, byrow = TRUE))
```

---

combinate2 *Like combinate but adds on to previous call*

Description

Like combinate but adds on to previous call

Usage

```
combinate2(X, Y)
```

Arguments

X	A list containing the elements Sspace (matrix), and logC (vector), the result of a call to combinate
Y	Matrix 2

Value

A list containing Sspace, the sample space (vectors of counts), and logC, a vector of the log multinomial coefficients.

Examples

```
slist_2_3 <- combinate(matrix(sspace_multinom(2, 5), ncol = 2, byrow = TRUE),
   matrix(sspace_multinom(3, 6), ncol = 3, byrow = TRUE))

sl_2_3_4 <- combinate2(slist_2_3, matrix(sspace_multinom(4, 3), ncol = 4, byrow = TRUE))
```

---

itp_root *Find a univariate root of the function f*

Description

This finds the value $x \in [a, b]$ such that $f(x) = 0$ using the one-dimensional root finding ITP method (Interpolate Truncate Project). Also see itp.

Usage

```
itp_root(
  f,
  a,
  b,
  k1 = 0.1,
  k2 = 2,
  n0 = 1,
  eps = 0.005,
  maxiter = 100,
  fa = NULL,
  fb = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`f`	The function to find the root of in terms of its first (one-dimensional) argument
`a`	The lower limit
`b`	The upper limit
`k1`	A tuning parameter
`k2`	Another tuning parameter
`n0`	Another tuning parameter
`eps`	Convergence tolerance
`maxiter`	Maximum number of iterations
`fa`	The value of f(a), if NULL then will be calculated
`fb`	The value of f(b), if NULL then will be calculated
`verbose`	Prints out information during iteration
`...`	Other arguments passed on to f

Value

A numeric vector of length 1, the root at the last iteration

References

I. F. D. Oliveira and R. H. C. Takahashi. 2020. An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality. ACM Trans. Math. Softw. 47, 1, Article 5 (March 2021), 24 pages. https://doi.org/10.1145/3423597

Examples

```
fpoly <- function(x) x^3 - x - 2 ## example from the ITP_method wikipedia entry
itp_root(fpoly, 1, 2, eps = .0001, verbose = TRUE)
```

log_multinom_coef *Calculate log of multinomial coefficient*

Description

Calculate log of multinomial coefficient

Usage

```
log_multinom_coef(x, sumx)
```

Arguments

x	Vector of observed counts in each cell
sumx	Total count

Value

The vector of log multinomial coefficients

Examples

```
S0 <- matrix(sspace_multinom(4, 6), ncol = 4, byrow = TRUE)
logC0<- apply(S0,1,log_multinom_coef,sumx=6)
```

pvalue_psi0 *Compute a p value for the test of psi <= psi0 (lower = TRUE) or psi >= psi0 (lower = FALSE)*

Description

Compute a p value for the test of psi <= psi0 (lower = TRUE) or psi >= psi0 (lower = FALSE)

Usage

```
pvalue_psi0(
  psi0,
  psi,
  psi_hat,
  psi_obs,
  alternative = "two.sided",
  maxit,
  chunksize,
  p_target,
  SSpacearr,
```

```
  logC,
  d_k,
  psi_is_vectorized = FALSE,
  theta_sampler = runif_dk_vects,
  ga = FALSE,
  ga_gfactor = 1,
  ga_lrate = 0.01,
  ga_restart_every = 10,
  warn = TRUE
)
```

Arguments

psi0	The null hypothesis value for the parameter being tested.
psi	Function that takes in parameters and outputs a real valued number for each parameter. Can be vectorized rowwise for a matrix or not.
psi_hat	The vector of psi values at each element of the sample space
psi_obs	The observed estimate at the given data
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less"
maxit	Maximum number of iterations of the Monte Carlo procedure
chunksize	The number of samples to take from the parameter space at each iteration
p_target	If a p-value is found that is greater than p_target, terminate the algorithm early.
SSpacearr	The sample space matrix
logC	The log multinomial coefficients for each row of the sample space
d_k	The vector of dimensions
psi_is_vectorized	Is psi vectorized by row?
theta_sampler	Function to take samples from the $Theta$ parameter space. Default is runif_dk_vects.
ga	Logical, if TRUE, uses gradient ascent.
ga_gfactor	Concentration parameter scale in the gradient ascent algorithm. A number or "adapt"
ga_lrate	The gradient ascent learning rate
ga_restart_every	Restart the gradient ascent after this number of iterations at a sample from
warn	If TRUE, will give a warning if no samples from the null space are found

Value

A vector with two p-values, one for the lower, and one for the greater

Examples

```
sspace_3_5 <- matrix(sspace_multinom(3, 5), ncol = 3, byrow = TRUE)
psi <- function(theta) max(theta)
logC <- apply(sspace_3_5, 1, log_multinom_coef, sumx = 5)
psi_hat <- apply(sspace_3_5, 1, \(x) psi(x / sum(x)))
pvalue_psi0(.3, psi, psi_hat, .4, maxit = 10, chunksize = 100,
 p_target = 1, SSpacearr = sspace_3_5, logC = logC, d_k = 3, warn = FALSE)
```

---

rdirich_dk_vects — *Sample independently from Dirichlet distributions for each of d_k vectors*

Description

Sample independently from Dirichlet distributions for each of d_k vectors

Usage

```
rdirich_dk_vects(nsamp, alpha)
```

Arguments

nsamp	number of samples to take
alpha	List of vectors of concentration parameters

Value

A matrix with sum(d_k) columns and nsamp rows

Examples

```
rdirich_dk_vects(10, list(rep(1, 3), rep(1, 4), rep(1, 2)))
```

---

runif_dk_vects — *Sample uniformly and independently from d_k simplices*

Description

Sample uniformly and independently from d_k simplices

Usage

```
runif_dk_vects(d_k, nsamp)
```

Arguments

`d_k`	vector of vector lengths
`nsamp`	number of samples to take

Value

A matrix with sum(d_k) columns and nsamp rows

Examples

```
runif_dk_vects(c(3, 4, 2), 10)
```

`sample_unit_simplexn` *Sample n times from the unit simplex in d dimensions*

Description

Sample n times from the unit simplex in d dimensions

Usage

```
sample_unit_simplexn(d, n)
```

Arguments

`d`	the dimension
`n`	the number of samples to take uniformly in the d space

Value

The grid over Theta, the parameter space. To be converted to a matrix with d columns and nsamp rows

Examples

```
matrix(sample_unit_simplexn(3, 10), ncol = 3, byrow = TRUE)
```

sspace_multinom — *Enumerate the multinomial sample space*

Description

Enumerate the multinomial sample space

Usage

```
sspace_multinom(d, n)
```

Arguments

d	The dimension
n	The sample size

Value

A vector enumerating the sample space, to be converted to a matrix with d columns and choose(n + d - 1, d - 1) rows

Examples

```
matrix(sspace_multinom(3, 5), ncol = 3, byrow = TRUE)
```

sspace_multinom_slow — *Enumerate the sample space of a multinomial*

Description

We have d mutually exclusive outcomes and n independent trials. This function enumerates all possible vectors of length d of counts of each outcome for n trials, i.e., the sample space. The result is output as a matrix with d columns where each row represents a possible observation. See sspace_multinom for a faster implementation using Rust.

Usage

```
sspace_multinom_slow(d, n)
```

Arguments

d	Dimension
n	Size

Value

A matrix with d columns

Examples

```
d4s <- sspace_multinom_slow(4, 8)
stopifnot(abs(sum(apply(d4s, 1, dmultinom, prob = rep(.25, 4))) - 1) < 1e-12)
```

`xactonomial` *Improved inference for a real-valued function of multinomial parameters*

Description

We consider the k sample multinomial problem where we observe k vectors (possibly of different lengths), each representing an independent sample from a multinomial. For a given function psi which takes in the concatenated vector of multinomial probabilities and outputs a real number, we are interested in computing a p-value for a test of psi >= psi0, and constructing a confidence interval for psi.

Usage

```
xactonomial(
  data,
  psi,
  statistic = NULL,
  psi0 = NULL,
  alternative = c("two.sided", "less", "greater"),
  psi_limits,
  theta_null_points = NULL,
  p_target = 1,
  conf_int = TRUE,
  conf_level = 0.95,
  itp_maxit = 10,
  itp_eps = 0.005,
  p_value_limits = NULL,
  maxit = 50,
  chunksize = 500,
  theta_sampler = runif_dk_vects,
  ga = TRUE,
  ga_gfactor = "adapt",
  ga_lrate = 0.01,
  ga_restart_every = 10
)
```

Arguments

`data`	A list with k elements representing the vectors of counts of a k-sample multinomial
`psi`	Function that takes in parameters and outputs a real valued number for each parameter. Can be vectorized rowwise for a matrix or not.
`statistic`	Function that takes in a matrix with data vectors in the rows, and outputs a vector with the number of rows in the matrix. If NULL, will be inferred from psi by plugging in the empirical proportions.
`psi0`	The null hypothesis value for the parameter being tested.
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less"
`psi_limits`	A vector of length 2 giving the lower and upper limits of the range of $\psi(\theta)$
`theta_null_points`	An optional matrix where each row is a theta value that gives psi(theta) = psi0. If this is supplied and psi0 = one of the boundary points, then a truly exact p-value will be calculated.
`p_target`	If a p-value is found that is greater than p_target, terminate the algorithm early.
`conf_int`	If TRUE, calculates a confidence interval by inverting the p-value function
`conf_level`	A number between 0 and 1, the confidence level.
`itp_maxit`	Maximum iterations to use in the ITP algorithm. Only relevant if conf_int = TRUE.
`itp_eps`	Epsilon value to use for the ITP algorithm. Only relevant if conf_int = TRUE.
`p_value_limits`	A vector of length 2 giving lower bounds on the p-values corresponding to psi0 at psi_limits. Only relvant if conf_int = TRUE.
`maxit`	Maximum number of iterations of the Monte Carlo procedure
`chunksize`	The number of samples to take from the parameter space at each iteration
`theta_sampler`	Function to take samples from the $Theta$ parameter space. Default is runif_dk_vects.
`ga`	Logical, if TRUE, uses gradient ascent.
`ga_gfactor`	Concentration parameter scale in the gradient ascent algorithm. A number or "adapt"
`ga_lrate`	The gradient ascent learning rate
`ga_restart_every`	Restart the gradient ascent after this number of iterations at a sample from `theta_sampler`

Details

Let T_j be distributed Multinomial$_{d_j}(\boldsymbol{\theta}_j, n_j)$ for $j = 1, \ldots, k$ and denote $\boldsymbol{T} = (T_1, \ldots, T_k)$ and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$. The subscript d_j denotes the dimension of the multinomial. Suppose one is interested in the parameter $\psi = \tau(\boldsymbol{\theta}) \in \Psi \subseteq \mathbb{R}$. Given a sample of size n from $\boldsymbol{T}$, say $\boldsymbol{X} = (X_1, \ldots, X_k)$, which is a vector of counts obtained by concatenating the k independent count vectors, let $G(\boldsymbol{X})$ denote a real-valued statistic that defines the ordering of the sample space.

Tne default choice of the statistic is to estimate $\boldsymbol{\theta}$ with the sample proportions and plug them into $\tau(\boldsymbol{\theta})$. This function calculates a p value for a test of the null hypothesis $H_0 : \psi(\boldsymbol{\theta}) \neq \psi_0$ for the two sided case, $H_0 : \psi(\boldsymbol{\theta}) \leq \psi_0$ for the case `alternative = "greater"`, and $H_0 : \psi(\boldsymbol{\theta}) \geq \psi_0$ for the case `alternative = "less"`. We make no assumptions and do not rely on large sample approximations. It also optionally constructs a $1 - \alpha$ percent confidence interval for ψ. The computation is somewhat involved so it is best for small sample sizes. The calculation is done by sampling a large number of points from the null parameter space Θ_0, then computing multinomial probabilities under those values for the range of the sample space where the statistic is as or more extreme than the observed statistic given data. It is basically the definition of a p-value implemented with Monte Carlo methods. Some options for speeding up the calculation are available.

Value

An object of class "htest", which is a list with the following elements:

estimate The value of the statistic at the observed data

p.value The p value

conf.int The upper and lower confidence limits

null.value The null hypothesis value provided by the user

alternative The type of test

method A description of the method

data.name The name of the data object provided by the user

p.sequence A list with two elements, p.null and p.alt containing the vector of p values at each iteration for the less than null and the greater than null. Used for assessing convergence.

Specifying the function psi

The psi parameter should be a function that either: 1) takes a vector of length sum(d_j) (the total number of bins) and outputs a single number, or 2) takes a matrix with number of columns equal to sum(d_j), and arbitrary number of rows and outputs a vector with length equal to the number of rows. In other words, psi can be not vectorized or it can be vectorized by row. Writing it so that it is vectorized can speed up the calculation. See examples.

Boundary issues

It is required to provide psi_limits, a vector of length 2 giving the smallest and largest possible values that the function psi can take, e.g., `c(0, 1)`. If the null hypothesis value psi0 is at one of the limits, it is often the case that sampling from the null parameter space is impossible because it is a set of measure 0. While it may have measure 0, it is not empty, and will contain a finite set of points. Thus you should provide the argument `theta_null_points` which is a matrix where the rows contain the finite set (sometimes 1) of points θ such that $\tau(\theta) = \psi_0$. There is also an argument called `p_value_limits` that can be used to improve performance of confidence intervals around the boundary. This should be a vector of length 2 with the p-value for a test of psi_0 <= psi_limits[1] and the p-value for a test of psi_0 >= psi_limits[2]. See examples.

Optimization options

For p-value calculation, you can provide a parameter p_target, so that the sampling algorithm terminates when a p-value is found that exceeds p_target. The algorithm begins by sampling uniformly from the unit simplices defining the parameter space, but alternatives can be specified in `theta_sampler`. By default gradient ascent (`ga = TRUE`) is performed during the p-value maximization procedure, and `ga_gfactor` and `ga_lrate` control options for the gradient ascent. At each iteration, the gradient of the multinomial probability at the current maximum theta is computed, and a step is taken to `theta + lrate * gradient`. Then for the next iteration, a set of `chunksize` samples are drawn from a Dirichlet distribution with parameter `ga_gfactor * (theta + ga_lrate * gradient)`. If `ga_gfactor = "adapt"` then it is set to `1 / max(theta)` at each iteration. The ITP algorithm itp_root is used to find roots of the p-value function as a function of the psi0 value to get confidence intervals. The maximum number of iterations and epsilon can be controlled via `itp_maxit, itp_eps`.

References

Sachs, M.C., Gabriel, E.E. and Fay, M.P., 2024. Exact confidence intervals for functions of parameters in the k-sample multinomial problem. arXiv preprint arXiv:2406.19141.

Examples

```
psi_ba <- function(theta) {
  theta1 <- theta[1:4]
  theta2 <- theta[5:8]
  sum(sqrt(theta1 * theta2))
  }
data <- list(T1 = c(2,1,2,1), T2 = c(0,1,3,3))
xactonomial(data, psi_ba, psi_limits = c(0, 1), psi0 = .5,
  conf_int = FALSE, maxit = 15, chunksize = 200)

# vectorized by row
psi_ba_v <- function(theta) {
theta1 <- theta[,1:4, drop = FALSE]
theta2 <- theta[,5:8, drop = FALSE]
rowSums(sqrt(theta1 * theta2))
}
data <- list(T1 = c(2,1,2,1), T2 = c(0,1,3,3))
xactonomial(data, psi_ba_v, psi_limits = c(0, 1), psi0 = .5,
 conf_int = FALSE, maxit = 10, chunksize = 200)

 # example of using theta_null_points
 # psi = 1/3 occurs when all probs = 1/3
 psi_max <- function(pp) {
   max(pp)
 }

data <- list(c(13, 24, 13))

xactonomial(data, psi_max, psi_limits = c(1 / 3, 1), psi0 = 1/ 3,
  conf_int = FALSE, theta_null_points = t(c(1/3, 1/3, 1/3)))
```

```
## in this case using p_value_limits improves confidence interval performance

 xactonomial(data, psi_max, psi_limits = c(1 / 3, 1), psi0 = 1/ 3,
  conf_int = TRUE, theta_null_points = t(c(1/3, 1/3, 1/3)),
  p_value_limits = c(.1, 1e-8))
```

Index