Package: perms (via r-universe)

August 20, 2024

Type Package

Title Fast Permutation Computation

Version 1.14

Date 2024-08-17

Description Implements the algorithm of Christensen (2024) <doi:10.1214/22-BA1353> for estimating marginal likelihoods via permutation counting.

License BSD_2_clause + file LICENSE

Copyright Yann Collet in xxhash.h and xxhash.c

Encoding UTF-8

RoxygenNote 7.3.1

Imports Rdpack

Depends foreach, doParallel, parallel

RdMacros Rdpack

NeedsCompilation yes

Author Per August Jarval Moen [cre, aut] (<https://orcid.org/0009-0003-9990-8341>), Dennis Christensen [aut] (<https://orcid.org/0000-0002-7540-7695>), Yann Collet [cph]

Maintainer Per August Jarval Moen <pamoen@math.uio.no>

Repository CRAN

Date/Publication 2024-08-17 14:40:02 UTC

Contents

get_log_ML *get_log_ML*

Description

Computes the log marginal likelihood of the data from the log permanents. Given the computed log permanents log_perms, this function computes the log marginal likelihood using the formula (2.3) in [1]. It is assumed that there are no repeated trials. If the data contain repeated trials, then the appropriate log binomial factor must be added to the output of this function.

Usage

```
get_log_ML(log_perms, n, debug = FALSE)
```

Arguments

log_perms	A vector length n containing the computed log permanents, where a zero permanent is indicated by a NA value.
n	Sample size.
debug	If TRUE, debug information is printed.

Value

The estimated log marginal likelihood. A NA value is returned if there are no non-zero numbers.

References

[1] Christensen, D (2024). Inference for Bayesian nonparametric models with binary response data via permutation counting. Bayesian Analysis, DOI: 10.1214/22-BA1353.

Examples

```
library(perms)
set.seed(1996)
n = 100
t = seq(0, 1, length.out=n)
y = c(rep(0, n/2), rep(1, n/2))
S = 200
X = matrix(runif(n*S),nrow = S, ncol = n)

log_perms = get_log_perms(X, t, y, debug = FALSE, parallel = FALSE, num_cores = NULL)

num_nonzero_perms = sum(!is.na(log_perms))
num_nonzero_perms

log_ML = get_log_ML(log_perms, n, FALSE)
log_ML
```

get_log_ML_bioassay *get_log_ML_bioassay*

Description

Computes the log marginal likelihood of bioassay data from the log permanents. Given the computed log permanents log_perms, this function computes the log marginal likelihood using the formula (2.3) in [1]. It takes care of repeated trials by adding the appropriate log binomial factor.

Usage

```
get_log_ML_bioassay(log_perms, successes, trials, debug = FALSE)
```

Arguments

log_perms	A vector length n containing the computed log permanents, where a zero permanent is indicated by a NA value.
successes	A vector of length n contatining the number of successful trials at each level.
trials	A vector of length n containing the number of trials at each level.
debug	If TRUE, debug information is printed.

Value

The estimated log marginal likelihood. A NA value is returned if there are no non-zero numbers.

References

[1] Christensen, D (2024). Inference for Bayesian nonparametric models with binary response data via permutation counting. Bayesian Analysis, DOI: 10.1214/22-BA1353.

Examples

```
## Dirichlet toy model
library(perms)
set.seed(1996)
n = 500
num_trials = 10
levels = seq(-1, 1, length.out = num_trials)

trials = rep(n %/% num_trials, num_trials)
successes = c(10, 26, 10, 20, 20, 19, 29, 24, 31, 33)

S = 200
alpha = 1.0

get_X = function(S,n,alpha,seed){
  set.seed(seed)
  X = matrix(0, nrow = S, ncol = n)
```

```
  for (s in 1:S) {
    X[s,1] = rnorm(1)
    for (i in 2:n) {
      u = runif(1)
      if(u < (alpha/(alpha+i-1))){
        X[s,i] = rnorm(1)
      }else{
        if(i==2){
          X[s,i] = X[s,1]
        }else{
          X[s,i] = sample(X[s, 1:(i-1)],size=1)
        }
      }

    }

  }
  return(X)
}

seed = 1996
X = get_X(S, n, alpha, seed)
log_perms = get_log_perms_bioassay(X, levels, successes, trials,
           debug=FALSE,parallel = FALSE)
log_ml = get_log_ML_bioassay(log_perms, successes, trials)

proportion = sum(!is.na(log_perms)) / S*100

proportion
log_ml
```

`get_log_perms` *get_log_perms*

Description

Computes log permanents associated with simulated latent variables. Each row of the S x n matrix X contains a random sample of size n from the data model. If there is only a single covariate, then the observed data are represented as (t,y), where t is the observed values of the covariate and y is the vector of indicator variables. If there are more covariates or the problem is phrased as binary classification (see Section 5 in [1]), then t is an S x n matrix since the threshold values change in each iteration. The function returns a vector of log permanents corresponding to each sample in X.

Usage

```
get_log_perms(X, tt, y, debug = FALSE, parallel = TRUE, num_cores = NULL)
```

Arguments

X	A matrix of dimension S x n, in which each row contains a sample from the data model.
tt	Either: A vector of length n containing the observed values of the covariate, Or: A matrix of dimension S x n (if there are several covariates).
y	A vector of length n indicating whether x_i<=t_i for each i in the observed data.
debug	If TRUE, debug information is printed.
parallel	If TRUE, computation is run on several cores
num_cores	(Optional) Specifies the number of cores to use if parallel = TRUE

Value

Vector of log permanents,each element associated to the corresponding row in X. A zero valued permanent is indicated by a NA value.

References

[1] Christensen, D (2024). Inference for Bayesian nonparametric models with binary response data via permutation counting. Bayesian Analysis, DOI: 10.1214/22-BA1353.

Examples

```
library(perms)
set.seed(1996)
n = 100
t = seq(0, 1, length.out=n)
y = c(rep(0, n/2), rep(1, n/2))
S = 200
X = matrix(runif(n*S),nrow = S, ncol = n)

log_perms = get_log_perms(X, t, y, debug = FALSE, parallel = FALSE, num_cores = NULL)

num_nonzero_perms = sum(!is.na(log_perms))
num_nonzero_perms

log_ML = get_log_ML(log_perms, n, FALSE)
log_ML
```

Description

Computes log permanents associated with simulated latent variables X with bioassay data. Each row of the matrix X contains a random sample of size n from the data model. The observed data are represented as (levels, successes, trials), where levels are the different levels at which trials were conducted, successes is the vector of the number of successes per level, and trials is the vector of the total number of trials per level. The function returns a vector of log permanents corresponding to each sample. Note that n must be equal to the sum of the entries of trials.

Usage

```
get_log_perms_bioassay(
  X,
  levels,
  successes,
  trials,
  debug = FALSE,
  parallel = TRUE,
  num_cores = NULL
)
```

Arguments

X	A matrix of dimension S x n, in which each row contains a sample from the data model.
levels	A vector containing the levels at which trials were conducted.
successes	A vector containing the number of successful trials at each level.
trials	A vector containing the number of trials at each level.
debug	If TRUE, debug information is printed.
parallel	If TRUE, computation is run on several cores
num_cores	(Optional) Specifies the number of cores to use if parallel = TRUE

Value

Vector of log permanents, each element associated to the corresponding row in X. A zero valued permanent is indicated by a NA value.

References

[1] Christensen, D (2024). Inference for Bayesian nonparametric models with binary response data via permutation counting. Bayesian Analysis, DOI: 10.1214/22-BA1353.

Examples

```
## Dirichlet toy model
library(perms)
set.seed(1996)
n = 500
num_trials = 10
```

```
levels = seq(-1, 1, length.out = num_trials)

trials = rep(n %/% num_trials, num_trials)
successes = c(10, 26, 10, 20, 20, 19, 29, 24, 31, 33)

S = 200
alpha = 1.0

get_X = function(S,n,alpha,seed){
  set.seed(seed)
  X = matrix(0, nrow = S, ncol = n)
  for (s in 1:S) {
    X[s,1] = rnorm(1)
    for (i in 2:n) {
      u = runif(1)
      if(u < (alpha/(alpha+i-1))){
        X[s,i] = rnorm(1)
      }else{
        if(i==2){
          X[s,i] = X[s,1]
        }else{
          X[s,i] = sample(X[s, 1:(i-1)],size=1)
        }
      }

    }

  }
  return(X)
}

seed = 1996
X = get_X(S, n, alpha, seed)
log_perms = get_log_perms_bioassay(X, levels, successes, trials,
           debug=FALSE,parallel = FALSE)
proportion = sum(!is.na(log_perms)) / S*100

proportion
```

`log_sum_exp` *log_sum_exp*

Description

Computes the log sum exp of a vector. Given input array = [x_1, ..., x_n], returns x_* + log(exp(x_1 - x_*) + ... + exp(x_n - x_*)), where x_* = max(x_1, ... x_n). Ignores entries with NA value.

Usage

```
log_sum_exp(x)
```

Arguments

x	Input vector.

Value

The log-sum-exp of the entries of the input vector.

Examples

```
library(perms)
x = c(1,2,3,-1,-1,1)
log_sum_exp(x)
```

Index