Package ‘dupiR’
March 21, 2024

Type Package
Title Bayesian Inference from Count Data using Discrete Uniform Priors
Version 1.2.1
Date 2024-03-17
Depends R (>= 2.15.1), methods
Author Federico Comoglio [aut, cre],
Maurizio Rinaldi [aut]
Maintainer Federico Comoglio <federico.comoglio@gmail.com>
Description We consider a set of sample counts obtained by sampling arbitrary fractions of a finite volume containing an homogeneously dispersed population of identical objects. This package implements a Bayesian derivation of the posterior probability distribution of the population size using a binomial likelihood and non-conjugate, discrete uniform priors under sampling with or without replacement. This can be used for a variety of statistical problems involving absolute quantification under uncertainty. See Comoglio et al. (2013) <doi:10.1371/journal.pone.0074388>.
License GPL-2
LazyLoad yes
RoxygenNote 7.3.1
Encoding UTF-8
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
Imports graphics, plotrix, stats, utils
NeedsCompilation no
Repository CRAN
Date/Publication 2024-03-21 16:20:05 UTC

R topics documented:

  dupiR-package  ................................................................. 2
  compute_ecdf  ................................................................. 3
Bayesian inference from count data using discrete uniform priors

Description

This package allows to infer population sizes using a binomial likelihood and least informative discrete uniform priors.

Author(s)

Federico Comoglio <federico.comoglio@gmail.com>
Maurizio Rinaldi

References

**compute_ecdf**

*Compute ECDF (empirical cumulative distribution function)*

**Description**

Compute ECDF (empirical cumulative distribution function)

**Usage**

```r
compute_ecdf(posterior)
```

**Arguments**

- `posterior` numeric vector of posterior probabilities over the prior support

**Value**

numeric vector with empirical cumulative distribution function (cumulative sum of posterior)

---

**compute_normalization_constant**

*Compute normalization constant*

**Description**

Compute normalization constant

**Usage**

```r
compute_normalization_constant(counts, n_start, n_end, f_product)
```

**Arguments**

- `counts` integer vector of counts
- `n_start` start of prior support range
- `n_end` end of prior support range
- `f_product` product of (1-fractions)

**Value**

normalization constant to compute posterior density
compute_posterior

Compute the posterior probability distribution of the population size for an object of class Counts

Description

Compute the posterior probability distribution of the population size using a discrete uniform prior and a binomial likelihood ("dup" algorithm, Comoglio et al.). An approximation using a Gamma prior and a Poisson likelihood is used when applicable ("gamma" algorithm) method (see Clough et al. for details)

Usage

compute_posterior(
  object,
  n_start,
  n_end,
  replacement = FALSE,
  b = 1e-10,
  alg = "dup"
)

Arguments

object object of class Counts
n_start start of prior support range
n_end end of prior support range
replacement was sampling performed with replacement? Default to FALSE
b prior rate parameter of the gamma distribution used to compute the posterior with Clough. Default to 1e-10
alg algorithm to be used to compute posterior. One of ... . Default to "dup"

Value

an object of class Counts

Author(s)

Federico Comoglio

References


**Examples**

```r
counts <- new_counts(counts = c(20, 30), fractions = c(0.075, 0.10))

# default parameters ("dup" algorithm, sampling without replacement, default prior support)
posterior <- compute_posterior(counts)

# custom prior support ("dup" algorithm)
posterior <- compute_posterior(counts, n_start = 0, n_end = 1e3)

# gamma prior ("gamma" algorithm)
posterior <- compute_posterior(counts, alg = "gamma")

# sampling with replacement
posterior <- compute_posterior(counts, replacement = TRUE)
```

---

**compute_posterior_with_replacement**

*Compute posterior probability with replacement*

**Description**

Compute posterior probability with replacement

**Usage**

```r
compute_posterior_with_replacement(n, counts, f_product, denominator)
```

**Arguments**

- `n` integer for which to compute the posterior
- `counts` integer vector of counts
- `f_product` product of (1-fractions)
- `denominator` normalization constant returned by `compute_normalization_constant`

**Value**

posterior probability of `n`

**See Also**

`compute_normalization_constant`
compute_sum

Description
Compute sum of terms (function F, Comoglio et al.)

Usage
compute_sum(counts, n, f_product)

Arguments
- counts: integer vector of counts
- n: number of objects
- f_product: product of (1-fractions)

Value
sum of terms in function F

compute_term

Description
Compute single term (function F, Comoglio et al.)

Usage
compute_term(counts, n, f_product, t)

Arguments
- counts: integer vector of counts
- n: number of objects
- f_product: product of (1-fractions)
- t: index vector

Value
single term of function F
Counts-class

An S4 class to store measurements (count data, sampling fractions), prior support and posterior parameters

Description

An S4 class to store measurements (count data, sampling fractions), prior support and posterior parameters

Usage

## S4 method for signature 'Counts'
get_counts(object)

## S4 method for signature 'Counts'
get_fractions(object)

## S4 replacement method for signature 'Counts'
set_counts(object) <- value

## S4 replacement method for signature 'Counts'
set_fractions(object) <- value

## S4 method for signature 'Counts'
compute_posterior(
  object,
  n_start,
  n_end,
  replacement = FALSE,
  b = 1e-10,
  alg = "dup"
)

## S4 method for signature 'Counts'
get_posterior_param(object, low = 0.025, up = 0.975, ...)

## S4 method for signature 'Counts'
plot_posterior(object, low = 0.025, up = 0.975, xlab, step, ...)

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>object of class Counts</td>
</tr>
<tr>
<td>value</td>
<td>numeric vector of sampling fractions</td>
</tr>
<tr>
<td>n_start</td>
<td>start of prior support range</td>
</tr>
<tr>
<td>n_end</td>
<td>end of prior support range</td>
</tr>
<tr>
<td>replacement</td>
<td>was sampling performed with replacement? Default to FALSE</td>
</tr>
</tbody>
</table>
Counts-class

b  prior rate parameter of the gamma distribution used to compute the posterior with Clough. Default to 1e-10

alg  algorithm to be used to compute posterior. One of ... . Default to "dup"

low  1 - right tail posterior probability

up  left tail posterior probability

...  additional parameters to be passed to curve

xlab  x-axis label. Default to 'n' (no label)

step  integer defining the increment for x-axis labels (distance between two consecutive tick marks)

Value

counts vector from a Counts object

fractions vector from a Counts object

an object of class Counts

an object of class Counts

an object of class Counts

an object of class Counts

no return value, called for side effects

Methods (by generic)

• get_counts(Counts): Returns counts from a Counts object

• get_fractions(Counts): Returns fractions from a Counts object

• set_counts(Counts) <- value: Replaces counts of a Counts object with the provided values

• set_fractions(Counts) <- value: Replaces fractions of a Counts object with the provided values

• compute_posterior(Counts): Compute the posterior probability distribution of the population size

• get_posterior_param(Counts): Extract statistical parameters (e.g. credible intervals) from a posterior probability distribution

• plot_posterior(Counts): Plot posterior probability distribution and posterior parameters

Slots

counts integer vector of counts (required)

fractions numeric vector of sampling fractions (required)

n_start start of prior support range. If omitted and total counts greater than zero, computed as 0.5 * mle, where mle is the maximum likelihood estimate of the population size

n_end end of prior support range. If omitted and total counts greater than zero, computed as 2 * mle, where mle is the maximum likelihood estimate of the population size

f_product product of (1-fractions)
counts-class

mle  maximum likelihood estimate of the population size (ratio between total counts and total sampling fraction)
norm_constant  normalization constant
posterior  numeric vector of posterior probabilities over the prior support
map_p  maximum of posterior probability
map_index  index of prior support corresponding to the maximum a posteriori
map  maximum a posteriori of population size
q_low  lower bound of the credible interval
q_low_p  probability of the lower bound of the credible interval
q_low_index  index of the prior support corresponding to q_low
q_low_cum_p  cumulative posterior probability from n_start to q_low (left tail)
q_up  upper bound of the credible interval
q_up_p  probability of the upper bound of the credible interval
q_up_index  index of the prior support corresponding to q_high
q_up_cum_p  cumulative posterior probability from q_high to n_end (right tail)
gamma  logical, TRUE if posterior computed using a Gamma approximation

Note

The posterior slot contains either the PMF or a logical value used to compute posterior parameters with a Gamma approximation (see reference for details)
Lower and upper bounds of the credible interval are computed at a default confidence level of 95
For more details on the normalization constant, see Corollary 1 in reference

Author(s)

Federico Comoglio

References


See Also

compute_posterior, get_posterior_param

Examples

# constructor:
# create an object of class 'Counts'
new_counts(counts = c(30, 35), fractions = c(0.075, 0.1))

# same, using new
new("Counts", counts = c(30, 35), fractions = c(0.075, 0.1))
gamma_poisson_clough: Compute posterior probability using a Gamma-Poisson model
(Clough et al.)

Description
Compute posterior probability using a Gamma-Poisson model (Clough et al.)

Usage
gamma_poisson_clough(object, n_start, n_end, a = 1, b = 1e-10)

Arguments
- object: object of class Counts
- n_start: start of prior support range
- n_end: end of prior support range
- a: prior shape parameter of the gamma distribution used to compute the posterior with Clough. Default to 1
- b: prior rate parameter of the gamma distribution used to compute the posterior with Clough. Default to 1e-10

Value
vector of posterior probabilities

Note
if support range spans more than 100k values, the posterior is not computed

get_counts: Get counts slot for an object of class Counts

Description
Get counts slot for an object of class Counts

Usage
get_counts(object)

Arguments
- object: object of class Counts
**get_fractions**

Value

counts vector from a Counts object

---

**get_fractions**  
*Get fractions slot for an object of class Counts*

**Description**

Get fractions slot for an object of class Counts

**Usage**

get_fractions(object)

**Arguments**

object  
object of class Counts

**Value**

fractions vector from a Counts object

---

**get_posterior_param**  
*Compute posterior probability distribution parameters (e.g. credible intervals) for an object of class Counts*

**Description**

This function computes posterior parameters and credible intervals at the given confidence level (default to 95%).

**Usage**

get_posterior_param(object, low = 0.025, up = 0.975, ...)

**Arguments**

object  
object of class Counts

low  
1 - right tail posterior probability

up  
left tail posterior probability

...  
additional parameters to be passed to `plot_posterior`

**Value**

an object of class Counts
**Author(s)**
Federico Comoglio

**References**


**Examples**
```r
counts <- new_counts(counts = c(20, 30), fractions = c(0.075, 0.10))
# default parameters ("dup" algorithm, sampling without replacement, default prior support)
posterior <- compute_posterior(counts)
get_posterior_param(posterior)
```

---

**initialize,Counts-method**

*Initialize Counts class*

**Description**
Initialize Counts class

**Usage**
```r
## S4 method for signature 'Counts'
initialize(.Object, counts, fractions)
```

**Arguments**
- `.Object` an object of class "Counts"
- `counts` integer vector of counts
- `fractions` numeric vector of sampling fractions
new_counts

Constructor for Counts class

Description

Constructor for Counts class

Usage

new_counts(counts, fractions)

Arguments

counts integer vector of counts
fractions numeric vector of sampling fractions

Value

An object of the Counts class

plot,Counts-method Plot method for Counts class

Description

Plot method for Counts class

Usage

## S4 method for signature 'Counts'
plot(x, y, ...)

Arguments

x object of class Counts
y none

... additional parameters to be passed to plot_posterior

Value

no return value, called for side effects
plot_posterior

Description
Plot posterior probability distribution and display posterior parameters for an object of class Counts

Usage
plot_posterior(object, low = 0.025, up = 0.975, xlab, step, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>object of class Counts</td>
</tr>
<tr>
<td>low</td>
<td>1 - right tail posterior probability</td>
</tr>
<tr>
<td>up</td>
<td>left tail posterior probability</td>
</tr>
<tr>
<td>xlab</td>
<td>x-axis label. Default to 'n' (no label)</td>
</tr>
<tr>
<td>step</td>
<td>integer defining the increment for x-axis labels (distance between two consecutive tick marks)</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters to be passed to curve</td>
</tr>
</tbody>
</table>

Value
no return value, called for side effects

Author(s)
Federico Comoglio

References

Examples

```r
counts <- new_counts(counts = c(20,30), fractions = c(0.075, 0.10))
# default parameters ("dup" algorithm, sampling without replacement, default prior support)
posterior <- compute_posterior(counts)

# plot posterior
plot_posterior(posterior, type = 'l', lwd = 3, col = 'blue3')
```
set_counts<-  

Set counts slot for an object of class Counts

Description
Set counts slot for an object of class Counts

Usage

set_counts(object) <- value

Arguments

object  object of class Counts
value    numeric vector of counts

Value
an object of class Counts

set_fractions<-  

Set fractions slot for an object of class Counts

Description
Set fractions slot for an object of class Counts

Usage

set_fractions(object) <- value

Arguments

object  object of class Counts
value    numeric vector of sampling fractions

Value
an object of class Counts
**show,Counts-method**  
*Print method for Counts class*

**Description**  
Print method for Counts class

**Usage**

```r
## S4 method for signature 'Counts'
show(object)
```

**Arguments**

- `object` object of class Counts

**Value**

no return value, called for side effects

---

**summary,Counts-method**  
*Summary method for Counts class*

**Description**  
Summary method for Counts class

**Usage**

```r
## S4 method for signature 'Counts'
summary(object, ...)
```

**Arguments**

- `object` object of class Counts
- `...` additional parameters affecting the summary produced

**Value**

no return value, called for side effects
# Index

* **class**
  - Counts-class, 7

* **package**
  - dupiR-package, 2
  - gamma_poisson_clough, 10
  - get_counts, 10
  - get_counts,Counts-method
    - (Counts-class), 7
  - get_fractions, 11
  - get_fractions,Counts-method
    - (Counts-class), 7
  - get_posterior_param, 9, 11
  - get_posterior_param,Counts-method
    - (Counts-class), 7
  - initialize,Counts-method, 12
  - new_counts, 13
  - plot,Counts-method, 13
  - plot_posterior, 11, 13, 14
  - plot_posterior,Counts-method
    - (Counts-class), 7
  - set_counts<-, 15
  - set_fractions<-, 15
  - set_fractions<-,Counts-method
    - (Counts-class), 7
  - show,Counts-method, 16
  - summary,Counts-method, 16

compute_ecdf, 3
compute_normalization_constant, 3, 5
compute_posterior, 4, 9
compute_posterior,Counts-method
  - (Counts-class), 7
compute_posterior_with_replacement, 5
compute_sum, 6
compute_term, 6
Counts-class, 7
curve, 8, 14
dupiR (dupiR-package), 2
dupiR-package, 2
gamma_poisson_clough, 10
get_counts, 10
get_counts,Counts-method
  - (Counts-class), 7
get_fractions, 11
get_fractions,Counts-method
  - (Counts-class), 7
get_posterior_param, 9, 11
get_posterior_param,Counts-method
  - (Counts-class), 7

initialize,Counts-method, 12
new_counts, 13
plot,Counts-method, 13
plot_posterior, 11, 13, 14
plot_posterior,Counts-method
  - (Counts-class), 7
set_counts<-, 15