Package ‘negenes’

December 8, 2025

Version 1.4

Date 2025-12-08

Title Estimating the Number of Essential Genes in a Genome

Author Karl W Broman [aut, cre] (ORCID: <`https://orcid.org/0000-0002-4914-6671`>)

Maintainer Karl W Broman <`broman@wisc.edu`>

Description Estimating the number of essential genes in a genome on the basis of data from a random transposon mutagenesis experiment, through the use of a Gibbs sampler.
Lamichhane et al. (2003) <doi:10.1073/pnas.1231432100>.

Depends R (>= 2.10.1)

Imports stats

Suggests roxygen2

License GPL (>= 3)

URL `https://github.com/kbroman/negenes`

BugReports `https://github.com/kbroman/negenes/issues`

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-12-08 12:50:02 UTC

Contents

`Mtb80` *Number of insertion sites in each gene in M tb CDC1551*

Description

Number of insertion sites in the initial 80% of each gene in the *Mycobacterium tuberculosis* CDC1551 genome.

Format

A matrix with two columns. Each row corresponds to a gene. (The row names are the MT numbers of the genes.) The element in the first column is the number of transposon insertion sites in the initial 80% that appear in the corresponding gene and in no other gene. The element in the second column is the number of transposon insertion sites in the initial 80% of both that gene and the following gene. There are 4204 rows; the 46 genes with no such site are not included.

Source

`https://www.jcvi.org/` (formerly TIGR)

References

Blades, N. J. and Broman, K. W. (2002) Estimating the number of essential genes in a genome by random transposon mutagenesis. Technical Report MS02-20,Department of Biostatistics, Johns Hopkins University, Baltimore, MD. `https://www.biostat.wisc.edu/~kbroman/publications/ms0220.pdf`

See Also

`negenes()`, `sim.mutants()`

Examples

```
## Not run: data(Mtb80)

# simulate 44% of genes to be essential
essential <- rep(0,nrow(Mtb80))
essential[sample(1:nrow(Mtb80),ceiling(nrow(Mtb80)*0.44))] <- 1

# simulate 759 mutants
counts <- sim.mutants(Mtb80[,1], essential, Mtb80[,2], 759)

# run the Gibbs sampler
output <- negenes(Mtb80[,1], counts[,1], Mtb80[,2], counts[,2])
## End(Not run)
```

`negenes` *Estimate the number of essential genes in a genome*

Description

Estimate, via a Gibbs sampler, the posterior distribution of the number of essential genes in a genome with data from a random transposon mutagenesis experiment. (See the technical report cited below.)

Usage

```
negenes(
  n.sites,
  counts,
  n.sites2 = NULL,
  counts2 = NULL,
  n.mcmc = 5000,
  skip = 49,
  burnin = 500,
  startp = 1,
  trace = TRUE,
  calc.prob = FALSE,
  return.output = FALSE
)
```

Arguments

`n.sites`	A vector specifying the number of transposon insertion sites in each gene (alone). All elements must by strictly positive.
`counts`	A vector specifying the number of mutants observed for each gene (alone). Must be the same length as `n.sites`, and all elements must be non-negative integers.
`n.sites2`	A vector specfying the number of transposon insertion sites shared by adjacent genes. The *i*th element is the number of insertion sites shared by genes *i* and *i*+1. The last element is for sites shared by genes *N* and 1. If NULL, assume all are 0.
`counts2`	A vector specfying the number of mutants shared by adjacent gene (analogous to `n.sites2`). The *i*th element is the number of mutants at sites shared by genes *i* and *i*+1. The last element is for sites shared by genes *N* and 1. If NULL, assume all are 0.
`n.mcmc`	Number of Gibbs steps to perform.
`skip`	An integer; only save every `skip` + 1st step.
`burnin`	Number of initial Gibbs steps to run (output discarded).
`startp`	Initial proportion of genes for which no mutant was observed that will be assumed essential for the Gibbs sampler. (Genes for which a mutant was observed are assumed non-essential; other genes are assumed essential independent with this probability.)

`trace`	If TRUE, print iteration number occassionally.
`calc.prob`	If TRUE, return the log posterior probability (up to an additive constant) for each saved iteration.
`return.output`	If TRUE, include detailed Gibbs results in the output.

Value

A list with components `n.essential` (containing the total number of essential genes at each iteration of the Gibbs sampler) `summary` (a vector containing the estimated mean, SD, 2.5 percentile and 97.5 percentile of the posterior distribution of the number of essential genes.

The next component, `geneprob`, is a vector with one element for each gene, containing the estimated posterior probability that each gene is essential. These are Rao-Blackwellized estimates.

If the argument `calc.prob` was true, there will also be a component `logprob` containing the log (base e) of the posterior probability (up to an additive constant) at each Gibbs step.

If the argument `return.output` was true, there will also be a matrix with `n.mcmc` / (`skip` + 1) rows (corresponding to the Gibbs steps) and a column for each gene The entries in the matrix are either 0 (essential gene) or 1 (non-essential gene) according to the state of that gene at that step in the Gibbs sampler.

Author(s)

Karl W Broman, `<broman@wisc.edu>`

References

- Blades, N. J. and Broman, K. W. (2002) Estimating the number of essential genes in a genome by random transposon mutagenesis. Technical Report MS02-20, Department of Biostatistics, Johns Hopkins University, Baltimore, MD. `https://www.biostat.wisc.edu/~kbroman/publications/ms0220.pdf`
- Lamichhane et al. (2003) A post-genomic method for predicting essential genes at subsaturation levels of mutagenesis: application to Mycobacterium, tuberculosis. Proc Natl Acad Sci USA 100:7213-7218 doi:10.1073/pnas.1231432100

See Also

`sim.mutants()`, `Mtb80()`

Examples

```
data(Mtb80)

# simulate 44% of genes to be essential
essential <- rep(0,nrow(Mtb80))
essential[sample(1:nrow(Mtb80),ceiling(nrow(Mtb80)*0.44))] <- 1

# simulate 759 mutants
counts <- sim.mutants(Mtb80[,1], essential, Mtb80[,2], 759)
```

```
# run the Gibbs sampler without returning detailed output
## Not run: output <- negenes(Mtb80[,1], counts[,1], Mtb80[,2], counts[,2])

# run the Gibbs sampler, returning the detailed output
## Not run: output2 <- negenes(Mtb80[,1], counts[,1], Mtb80[,2], counts[,2], return=TRUE)
```

`sim.mutants` *Simulate data for a random transposon mutagenesis experiment*

Description

Simulate data for a random transposon mutagenesis experiment.

Usage

```
sim.mutants(n.sites, essential, n.sites2 = NULL, n.mutants)
```

Arguments

`n.sites`	A vector specifying the number of transposon insertion sites in each gene. All elements must by strictly positive.
`essential`	A vector containing 1's (indicating that the corresponding gene is essential) and 0's (indicating that the corresponding gene is not essential). Must be the same length as `n.sites`.
`n.sites2`	A vector specfying the number of transposon insertion sites shared by adjacent genes. The *i*th element is the number of insertion sites shared by genes *i* and *i*+1. The last element is for sites shared by genes *N* and 1. If missing, these are assumed to be all 0.
`n.mutants`	Number of mutants to simulate.

Value

If `n.sites2` is missing or contains all 0's, a vector is returned containing the number of mutants observed for each gene.

If `n.sites2` is not missing and has some positive entries, a matrix with two columns is returned. The first column contains the number of mutants observed for each gene alone; the second column contains the number of mutants observed shared by adjacent genes.

Author(s)

Karl W Broman, `<broman@wisc.edu>`

References

Blades, N. J. and Broman, K. W. (2002) Estimating the number of essential genes in a genome by random transposon mutagenesis. Technical Report MS02-20, Department of Biostatistics, Johns Hopkins University, Baltimore, MD. https://www.biostat.wisc.edu/~kbroman/publications/ms0220.pdf

See Also

`negenes()`, `Mtb80()`

Examples

```
## Not run: data(Mtb80)

# simulate 44% of genes to be essential
essential <- rep(0,nrow(Mtb80))
essential[sample(1:nrow(Mtb80),ceiling(nrow(Mtb80)*0.44))] <- 1

# simulate 759 mutants
counts <- sim.mutants(Mtb80[,1], essential, Mtb80[,2], 759)

# run the Gibbs sampler
output <- negenes(Mtb80[,1], counts[,1], Mtb80[,2], counts[,2])
## End(Not run)
```

Index