Package ‘hbim’

May 9, 2022

Type Package
Title Hill/Bliss Independence Model for Combination Vaccines
Version 1.1
Date 2022-05-09
Author Michael P. Fay
Maintainer Michael P. Fay <mfay@niaid.nih.gov>
Depends R (>= 2.5.0), stats, mvtnorm
Description Calculate expected relative risk and proportion protected assuming normally distributed log10 transformed antibody dose for a several component vaccine. Uses Hill models for each component which are combined under Bliss indepen-dence. See Saul and Fay, 2007 <DOI:10.1371/journal.pone.0000850>.
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2022-05-09 14:40:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbim-package</td>
<td>2</td>
</tr>
<tr>
<td>calc.foldrange</td>
<td>2</td>
</tr>
<tr>
<td>deff.sigma</td>
<td>3</td>
</tr>
<tr>
<td>eff.mu</td>
<td>4</td>
</tr>
<tr>
<td>equiv.ab</td>
<td>7</td>
</tr>
<tr>
<td>equiv.increase</td>
<td>8</td>
</tr>
<tr>
<td>hbr</td>
<td>9</td>
</tr>
<tr>
<td>irdata</td>
<td>10</td>
</tr>
<tr>
<td>make.v</td>
<td>12</td>
</tr>
<tr>
<td>plotlogm.resp</td>
<td>12</td>
</tr>
<tr>
<td>refs</td>
<td>13</td>
</tr>
</tbody>
</table>

Index 14
Description

Calculate expected relative risk and proportion protected assuming normally distributed log10 transformed antibody dose for several component vaccine. Uses Hill models for each component which are combined under Bliss independence.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>hbim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2022-05-09</td>
</tr>
<tr>
<td>License:</td>
<td>GPL</td>
</tr>
</tbody>
</table>

The hbim package allows users to reproduce plots and calculations for Saul and Fay (2007). See vignette("hbimdetails").

Author(s)

M.P. Fay, Maintainer: Michael Fay <mfay@niaid.nih.gov>

References


Description

Given a confidence interval and sample size, we find the standard error assuming confidence limits are calculated on the log10 responses by either normal confidence limits or t-distribution confidence limits. The fold-range is also output by either methods.

Usage

calc.foldrange(n, lower, upper, conf.level = 0.95)
Arguments

- `n`: vector of sample size(s) used to create confidence intervals
- `lower`: vector of lower confidence limits
- `upper`: vector of upper confidence limits
- `conf.level`: confidence level, default=.95

Details

See vignette("hbimdetails")

Value

A vector (or matrix) with elements (or columns)

- `n`: sample size
- `lower`: lower confidence limit
- `upper`: upper confidence limit
- `s.byt`: standard deviation assuming confidence intervals calculated by t-distribution
- `s.byZ`: standard deviation assuming confidence intervals calculated by normal distribution
- `foldrange.byt`: fold-range assuming confidence intervals calculated by t-distribution
- `foldrange.byZ`: fold-range assuming confidence intervals calculated by normal distribution

Examples

```r
## sample size=43, lower cl=65, upper cl=85
calc.foldrange(43,65,85)
```

Description

These 6 data sets were calculated using the associated function. For example, `deff.sigma` was calculated with `eff.sigma`. The 3 data sets that begin with `deff`, give the expected efficacy for several values of `mu`. The 3 data sets that begin with `dpp` give the percent protected with several values of `mu`. The data sets that end in `.sigma` change for different values of `sigma`, and similarly for `.mu` and `.rho` (see `deff.sigma`).

Usage

```r
data(deff.sigma)
data(deff.mu)
data(deff.rho)
data(dpp.sigma)
data(dpp.mu)
data(dpp.rho)
```
Format

The format is: List of 8

- mu vector of different values of mean for log10 antibody
- out1 matrix of either expected efficacy or percent protected for 1 component model, rows correspond to mu, cols correspond to cparms
- out2 matrix of either expected efficacy or percent protected for 2 component model, rows correspond to mu, cols correspond to cparms
- out3 matrix of either expected efficacy or percent protected for 3 component model, rows correspond to mu, cols correspond to cparms
- col1 vector of colors for different cparms of 1 component model
- col2 vector of colors for different cparms of 2 component model
- col3 vector of colors for different cparms of 3 component model
- cparms vector parameters that change

Examples

```r
## here is the code that produces the 6 data sets, it takes about 25 hours to run
## so it is commented out here
#NSIM<-5*10^5
#SIGMAS.POWER<-c(9,65,5000)
#SIGMAS<-log10(SIGMAS.POWER)/(2*qnorm(.975))
#SCOLORS<-c("green","blue","red")
#FACTORS<-c(1/10, 1/3, 1/2, 1)
#FCOLORS<-c("red", "green", "blue", "black")
#RHOS<-c(-.5,-.25, 0, .25, 0.5, 0.75, 1)
#RCOLORS<- c("black","tan","yellow","blue", "green", "red", "black")
#set.seed(1234521)
#MU<-((-40:40)/10)
#deff.sigma<-eff.sigma(mu=MU, sigmas=SIGMAS, COLORS = SCOLORS, rho = 0)
#deff.muc<-eff.muc(mu=MU, factor = FACTORS, COLORS = FCOLORS, sigma = SIGMAS[2], rho = 0)
#deff.rho<-eff.rho(mu=MU, sigma = SIGMAS[2], rho = RHOS, COLORS =RCOLORS,simulate=TRUE,nsim=NSIM)
#set.seed(32401)
#dpp.sigma<-pp.sigma(MU,sigmas=SIGMAS,COLORS = SCOLORS, rho = 0,nsim=NSIM)
#set.seed(21345123)
#dpp.muc<-pp.muc(MU,factor = FACTORS, COLORS = FCOLORS, sigma = SIGMAS[2], rho = 0, nsim=NSIM)
#set.seed(435919)
#dpp.rho<-pp.rho(MU,sigma = SIGMAS[2], rho = RHOS, COLORS =RCOLORS,nsim=NSIM)
```
Description

These functions create the data sets used in the plots. The first part of the name denotes the output created. Thus, eff.sigma, eff.mu, eff.rho create efficacy values, while pp.sigma, pp.mu, pp.rho create percent protected values. The second part of the name is the parameter which is changed. For example, eff.sigma creates efficacy values for different values of sigma. See details for a more complete description. Default for eff. functions is integration, default for pp. functions is simulation.

Usage

```r
eff.sigma(mu, sigmas, COLORS = c("red", "green", "blue"),
    rho = 0, ...)
eff.mu(mu, factor = c(1/10, 1/3, 1/2, 1),
    COLORS = c("red", "green", "blue", "black"),
    sigma = 0.553, rho = 0, ...)
eff.rho(mu, sigma = 0.553, rho = c(0, 0.25, 0.5, 0.75, 1),
    COLORS = c("black", "blue", "green", "red", "black"), ...)
pp.sigma(mu, sigmas, COLORS = c("red", "green", "blue"),
    rho = 0, nsim = 10^5)
pp.mu(mu, factor = c(1/10, 1/3, 1/2, 1),
    COLORS = c("red", "green", "blue", "black"),
    sigma = 0.553, rho = 0, nsim = 10^5)
pp.rho(mu, sigma = 0.553, rho = c(0, 0.25, 0.5, 0.75, 1),
    COLORS = c("black", "blue", "green", "red", "black"),
    nsim = 10^5)
```

Arguments

- **mu**: a vector of values of the mean of the log10 antibody
- **factor**: a vector of values for defining the means of the second and third component (see details and warnings)
- **COLORS**: colors for the plots, the ith color corresponds to the ith value of the parameter which is changing
- **sigmas**: a vector of values of the standard deviation of the log10 antibody
- **sigma**: a single value for sigma
- **rho**: correlation vector (of length one for .sigma and .mu functions) of the log10 antibody, negative values not allowed
- **nsim**: number of simulations for hbpp function
- **...**: additional parameters may be added to the hbr function

Details

For eff.sigma and pp.sigma we change sigma over the one, two, and three component model. For eff.mu and pp.mu we change the mean over the two and three component model. For eff.mu and pp.mu the factor parameter is associated with each level of the second and third component. See vignette("hbimdetails") for details. For eff.rho and pp.rho we change the correlation over
the two and three component model; for the jth column of the out2 and out3 matrices, all correlations are given by jth level of factor. Because these calculations may take hours, we save the original calculations used in the paper as output data, deff.sigma, deff.mu, deff.rho, dpp.sigma, dpp.mu, and dpp.rho. These output data set may be accessed by the command data(). For example, to access deff.sigma type data(deff.sigma).

Value

A list with items

- `out1`: response matrix for one component model, ith row corresponds to mu[i] and jth column corresponds to the jth level of the parameter which is changing
- `col1`: colors corresponding to columns of out1
- `out2`: response matrix for two component model, ith row corresponds to mu[i] and jth column corresponds to the jth level of the parameter which is changing
- `col2`: colors corresponding to columns of out2
- `out3`: response matrix for three component model, ith row corresponds to mu[i] and jth column corresponds to the jth level of the parameter which is changing
- `col3`: colors corresponding to columns of out3
- `cparms`: input vector of parameter that changes, e.g., factor vector
- `sigma`: input sigma
- `rho`: input rho

Warning

Note to save computation time these functions do not check that all variance-covariance matrices used in the internal functions are positive definite. If you get an error message you do not understand check to see if the variance-covariance matrix is positive definite by checking the eigen values. For example, with sigma=1, rho=-.6, the 3 components model do not have a positive definite variance-covariance matrix because there is a negative eigenvalue (to see this run eigen(make.v(3,-.6,1)) ).

Author(s)

M.P. Fay

See Also

vignette("hbimdetails")
equiv.ab

Equivalent antibody calculations by Linear Interpolation

Description

This function inputs two antibody by response curves and outputs values needed for plots of equivalent antibody response. This is called by other functions (`plotresp.equiv, plotresp.mix`). It is not to be called directly. For that purpose use `equiv.increase`.

Usage

equiv.ab(effab1, ab1, effab2, ab2, npts = 100)

Arguments

effab1         vector of responses for antibody 1
ab1            vector of doses of antibody 1
effab2         vector of responses for antibody 2
ab2            vector of doses of antibody 2
npts           number of points used in some output

Details

The function uses the `approx` function to do linear interpolation and find the needed values.

Value

A list containing:

- `abpts` a vector of values of antibody dose
- `abpts10` antilog of abpts, i.e., abpts raised to tenth power
- `equiv.eff2` equivalent response of antibody 2
- `equiv.eff1` equivalent response of antibody 1
- `equiv.ab1` vector of antibody doses that correspond with equiv.eff1
- `x` equiv.ab1-abpts
- `y` equiv.eff1

See Also

equiv.increase
equiv.increase  
*Calculate equivalent increase from two dose-response curves*

**Description**

This function takes two curves defined by vectors of x and y values and calculates the equivalent increase in the x value at the response value for the first curve at e1.

**Usage**

```r
equiv.increase(x1, y1, x2, y2, e1, xlog = TRUE)
```

**Arguments**

- `x1`: x vector for first curve
- `y1`: y vector for first curve
- `x2`: x vector for second curve
- `y2`: y vector for second curve
- `e1`: vector of y responses of first curve for associating with output
- `xlog`: TRUE if x values are log transformed, changes the output

**Details**

The function repeatedly uses the `approx` function to do linear interpolation.

**Value**

A list with 5 components

- `a1`: vector of x values associated with e1 from first curve
- `e2`: vector of y values associated with a1 from the second curve
- `a2`: vector of x values associated with e2 from the second curve
- `e1`: input vector for e1
- `equiv.increase`: vector of equivalent increases associated with e1

**Examples**

```r
data(deff.sigma)
D<-deff.sigma
equiv.increase(D$mu,D$out1[,2],D$mu,D$out2[,2],.5)
```
**hbrr** Calculate expected relative risk or percent protected from Hill model with Bliss Independence

**Description**

Assuming that the log10 transformed doses are normally distributed, we calculate the expected relative risk (using hbrr) or percent protected (using hbpp) from the Hill model using Bliss Independence. Numeric integration is the default for up to three components for hbrr, while simulation is the default for two or three components for hbpp.

**Usage**

```
hbrr(mu, v, a = rep(1, length(mu)), simulate = FALSE, nsim = 10^4, ...)  
hbpp(mu, v, a = rep(1, length(mu)), rp = 0.1, simulate = FALSE, nsim = 10^5, ...)
```

**Arguments**

- `mu`: mean vector of the log10 dose
- `v`: variance matrix of the log10 dose
- `a`: vector of slope parameters in the Hill model, one for each component
- `simulate`: estimation by simulation (TRUE) or numeric integration (FALSE)
- `nsim`: number of simulations, ignored if simulate=FALSE
- `rp`: protection bound, an individual is protected if relative risk is greater than rp
- `...`: additional parameters to pass to the integrate function

**Details**

Although the package adapt can do multidimensional integration, we have written specific functions to do this for up to 3 dimensions. This allows faster and more accurate integration. The integration is done by repeated calls to the integrate function. The functions which do the actual integration or simulation are internal functions which are not intended to be called by the user. These internal functions are: for hbrr, when simulate=FALSE, the function calls one of either `hbrr.integrate1`, `hbrr.integrate2`, `hbrr.integrate2.rhoeq1`, `hbrr.integrate3`, or `hbrr.integrate3.rhoeq1` (for 1,2, or 3 component, with or without rho=1, taken from the size of the mu vector and dimension of the v matrix) and when simulation=TRUE it calls `hbrr.simulate`. Similar functions exist for hbpp; however, the `hbpp.integrate2` and `hbpp.integrate3` may have problems because of the discontinuity in the integration function. That is why for two or three component models `hbpp.simulate` is used by default.

**Value**

A numeric value of the expected relative risk or percent protected.
Author(s)
M.P. Fay

References

Examples

```r
## example of two dimensional integral
hbrr(c(.123,.432),matrix(c(1,.5,.5,1),2,2))
## faster but less accurate estimation by simulation
hbrr(c(.123,.432),matrix(c(1,.5,.5,1),2,2),simulate=TRUE,nsim=10^4)
```

### irdata

**Immune Response data**

Description
Data from literature.

Usage
`data(irdata)`

Format
A data frame with 574 observations on the following 16 variables.

- **RecordNum**: a numeric vector
- **Old.Reference**: a numeric vector
- **Reference**: a numeric vector
- **Vaccine.and.trial.group**: a factor with levels (Pentacel + Recombivax) then
Carrier for conjugate vaccines: a factor with levels CRM197, Diphtheria toxoid, OMPC, Tetanus protein, Tetanus toxoid.

Age in yrs at first vaccination: a factor with levels 0.12, 0.17, 0.17-0.5, 0.25, 0.5, 1, 1-12, 1-2, 1.5, 11-18, 12-15, 15-18, 15-70, 16-65, 17-72, 18-32, 18-39, 18-40, 18-50, 18-60, 18-64, 19-52, 19-56, 19-57, 19-64, 19-70, 19-83, 2, 2-5, 20-45, 20-60, 20-61, 21-60, 3, 3-4, 4-14, 40-70, 5-16, 65-83.

Dose schedule in weeks: a factor with levels 0, 0, 2, 0, 2, 4, 0, 2, 60, 0, 4, 260, 0, 4, 26, 0, 4, 0, 10, 0, 4, 26, 0, 4, 52, 0, 8, 0, 8, 0, 8, 52, 0, 52, 0, 104, 0, 6, 13, 0, 8, 0, 8, 16, 0, 8, 18, 0, 8, 18, 0, 18, 220, 0, 8, 18, 270, 0, 8, 18, 320, 0, 8, 18, 44, 0, 8, 18, 45, 0, 8, 18, 60, 0, 8, 18, 70, 0, 8, 18, 4, 12, 22, 0, 8, 18, 4, 12, 22, 0, 8, 18, 4, 12, 22, 0, 8, 39, 0, 9, 37.

Num Immunizations: a numeric vector.

Endpoint in weeks after first vaccine: a numeric vector.


Units: a factor with levels EL, U/mL, EU, HI, IU, mIU, ng, SBA, ug.

GMT: a numeric vector.

GMT.95.pct.interval.low.limit: a numeric vector.

GMT.95.pct.interval.high.limit: a numeric vector.

n: a numeric vector.


Source

See data(refs) for references.

Examples

data(irdata)
irdata[1,]
make.v  Make Exchangeable Variance Matrix

Description

Not to be called directly. Used by eff.sigma, eff.mu, eff.rho, pp.sigma, pp.mu, and pp.rho.

Usage

make.v(n, r, sig2)

Arguments

- **n**: dimension of variance matrix
- **r**: correlation
- **sig2**: variance

Value

An variance-covariance matrix, with all diagonal elements equal and all off diagonal elements equal.

plotlogm.resp  Plot Hill/Bliss Independence Model Data.

Description

These functions take data output calculated from the data generating functions (see details) and plot either: the mean of the log transformed antibody doses by the response (plotlogm.resp), equivalent increase in antibody plots (plotresp.equiv), or response of one component versus a mixture (for details see vignette("hbimdetails")).

Usage

plotlogm.resp(D, YLAB = "Efficacy", YLIM = c(0, 1),
              XLIM = c(-2, 2), TITLE="")
plotresp.equiv(D, XLIM = c(0, 1), YLIM = c(1, 100),
               RLAB = "Efficacy of", bounds= XLIM, TITLE="")
plotresp.mix(D, RLAB = "Efficacy of", XYLIM = c(0, 1), TITLE="")
Arguments

- **D**: data, see details
- **YLAB**: y label
- **YLIM**: range of y axis
- **XLIM**: range of x axis
- **RLAB**: response label, currently use only either "Efficacy of" or "% Protected by"
- **.bounds**: bounds on response of second antibody curve, see vignette("hbimdetails")
- **XYLIM**: range of both x and y axes
- **TITLE**: title of plot

Details

The following functions create data sets for plotting: `eff.sigma`, `eff.mu`, `eff.rho`, `pp.sigma`, `pp.mu`, `pp.rho`. These functions plot that data. For details see vignette("hbimdetails").

Value

Plots

---

<table>
<thead>
<tr>
<th>refs</th>
<th>Reference list</th>
</tr>
</thead>
</table>

Description

Each reference is one long character string. See `data(irdata)` for data from each reference.

Usage

`data(refs)`

Format

The format is: Factor w/ 50 levels (the 50 references)

Examples

`data(refs)`
`refs[1]`
Index

* arith
  equiv.increase, 8
* datagen
  eff.mu, 4
* datasets
  deff.sigma, 3
  irdata, 10
  refs, 13
* hplot
  plotlogm.resp, 12
* misc
  equiv.ab, 7
  make.v, 12
* models
  hbrr, 9
* package
  hbim (hbim-package), 2
  hbim-package, 2
  hbpp (hbrr), 9
  hbpp.integrate2, 9
  hbpp.integrate3, 9
  hbpp.simulate, 9
  hbrr, 5, 9
  hbrr.integrate1, 9
  hbrr.integrate2, 9
  hbrr.integrate2.rhoeq1, 9
  hbrr.integrate3, 9
  hbrr.integrate3.rhoeq1, 9
  hbrr.simulate, 9
  integrate, 9
  irdata, 10
  make.v, 12
  plotlogm.resp, 12
  plotresp.equiv, 7
  plotresp.equiv (plotlogm.resp), 12
  plotresp.mix, 7
  plotresp.mix (plotlogm.resp), 12
  pp.mu, 12, 13
  pp.mu (eff.mu), 4
  pp.rho, 12, 13
  pp.rho (eff.mu), 4
  pp.sigma, 12, 13
  pp.sigma (eff.mu), 4
  refs, 13

approx, 7, 8

calc.foldrange, 2
deff.mu, 6
deff.mu (deff.sigma), 3
deff.rho, 6
deff.rho (deff.sigma), 3
deff.sigma, 3, 6
dpp.mu, 6
dpp.mu (deff.sigma), 3
dpp.rho, 6
dpp.rho (deff.sigma), 3
dpp.sigma, 6
dpp.sigma (deff.sigma), 3
eff.mu, 4, 12, 13
eff.rho, 12, 13
eff.rho (eff.mu), 4
eff.sigma, 3, 12, 13
eff.sigma (eff.mu), 4
equiv.ab, 7
equiv.increase, 7, 8

hbim (hbim-package), 2
hbim-package, 2
hbpp (hbrr), 9
hbpp.integrate2, 9
hbpp.integrate3, 9
hbpp.simulate, 9
hbrr, 5, 9
hbrr.integrate1, 9
hbrr.integrate2, 9
hbrr.integrate2.rhoeq1, 9
hbrr.integrate3, 9
hbrr.integrate3.rhoeq1, 9
hbrr.simulate, 9
integrate, 9
irdata, 10
make.v, 12
plotlogm.resp, 12
plotresp.equiv, 7
plotresp.equiv (plotlogm.resp), 12
plotresp.mix, 7
plotresp.mix (plotlogm.resp), 12
pp.mu, 12, 13
pp.mu (eff.mu), 4
pp.rho, 12, 13
pp.rho (eff.mu), 4
pp.sigma, 12, 13
pp.sigma (eff.mu), 4
refs, 13