Package 'vacem'

July 2, 2014

Version 0.1-1

Date 2012-01-30

Title Vaccination Activities Coverage Estimation Model

Description A likelihood framework for estimating the effective coverage of vaccination programs using cross-sectional surveys combined with administrative data.

Author Justin Lessler <jlessler@jhsph.edu>, Jessica Metcalf <cmetcalf@@princeton.edu>

Maintainer Ken Cline <kcline@jhsph.edu>

Depends foreach

Suggests RUnit

License GPL (>= 2)

LazyLoad yes

Collate 'vacem.r' 'coverage.r' 'analysis.r' 'sample.r'

Repository CRAN

Date/Publication 2012-01-31 06:39:02

NeedsCompilation no

R topics documented:

`vacem-package` *Vaccination Activities Coverage Estimation Model*

Description

A likelihood framework for estimating the effective coverage of vaccination programs using cross-sectional surveys combined with administrative data.

Details

Package:	vacem
Type:	Package
Version:	0.1-1
Date:	2012-01-30
License:	GPL (>= 2)
LazyLoad:	yes

The performance of routine and supplemental immunization activities is usually measured by the administrative method: dividing the number of doses distributed by the size of the target population. This package implements a method that combines the administrative data with measurements from cross-sectional surveys of vaccine coverage, e.g. Demographic and Health Surveys (DHS), to provide an improved estimate of vaccination coverage.

Author(s)

Justin Lessler <jlessler@jhsph.edu> Jessica Metcalf <cmetcalf@princeton.edu>

Maintainer: Ken Cline <kcline@jhsph.edu>

References

Lessler J, Metcalf CJE, Grais RF, Luquero FJ, Cummings DAT, et al. (2011) Measuring the Performance of Vaccination Programs Using Cross-Sectional Surveys: A Likelihood Framework and Retrospective Analysis. PLoS Med 8(10): e1001110. doi:10.1371/journal.pmed.1001110 http://www.plosmedicine.org/article/info:doi/10.1371/journal.pmed.1001110

Examples

```
# TBD
```

`calc.camp.coverage` *Calculate the expected coverage of each campaign given...*

Description

Calculate the expected coverage of each campaign given the trans.rho and alpha.

Usage

```
calc.camp.coverage(camps, trans.rho, alpha)
```

Arguments

`camps`	TBD
`trans.rho`	TBD
`alpha`	TBD

Details

Corresponds to equation 5 in paper

Value

TBD

`constant.post.9mo` *Defines a simple cumulative distribution function (CDF) with...*

Description

Defines a simple cumulative distribution function (CDF) with a constant "hazard" rate @p lambda of routine vaccination starting at age 8.5 months.

Usage

```
constant.post.9mo(age, lambda)
```

Arguments

`age`	the age of the individual
`lambda`	the routine vaccination "hazard" rate

Details

The cumulative distribution function determines an individual's probabilty of receiving a routine vaccination. It is passed into the weighting calculation function (@codew.matrix) by various coverage analysis functions such as @code ll.coverage and @code mcmc.estimate . A user specifies, by name, which CDF to use when an analysis function is invoked, e.g. @code mcmc.estimate(..., routine.cdf = "const.post.9m", ...) . See the documentation for the respective analysis functions for details about which CDFs are supported.

Value

```
pexp( @p age - 8.5, @p lambda )
```

@see w.matrix @see ll.coverage @see mcmc.estimate

Examples

```
x <- 0:24 ; lambda <- 0.5
plot( x=x, y=constant.post.9mo(x,lambda) )

x <- 0:24 ; lambda <- seq( 1.0, 0.1, -0.2 )
plot( type="n", x=x, xlab="age", ylab="constant.post.9mo(age,lambda)",
xlim=c(min(x),max(x)), ylim=c(0,1),
xaxp=c(min(x),max(x),(max(x)-min(x))/2) )
legend( "bottomright", "lambda", lambda, inset=0.1,
fill=c(1:length(lambda)) )
for ( i in 1:length(lambda) ) {
lines( x=x, y=constant.post.9mo(x,lambda[i]), col=i )
}
```

DAYS.PER.MONTH *The average number of days per month.*

Description

The average number of days per month.

Details

The @c DAYS.PER.MONTH constant is used when converting age values from days to months.

E.vacc *Calculates the expected number of vaccinations for individuals.*

Description

Calculates the expected number of vaccinations for individuals.

Usage

```
E.vacc(z, w, v, N, alpha, rho, log=FALSE, cond.on.accessible=FALSE)
```

Arguments

z	the eligibility vector for the individual, i.e. @codez[j] = 1 if this individual is eligible for campaign @c j, otherwise @codez[j] = 0
w	the weight vector for the individual, i.e. @code w[k] gives the probability of the individual receiving the routine vaccination in year @c k
v	the doses vector providing the number of vaccine doses nominally distributed in each campaign
N	the population size vector specifying the number of people in the target age range for each campaign
alpha	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
rho	the proportion of the population (scalar) that can be vaccinated, i.e @code1 - rho is the @e inaccessible portion of the general population.
log	a flag indicating whether the log probability should be returned; the default is @c FALSE NOTE: the @c log parameter is @b not currently used.
cond.on.accessible	a flag indicating whether the return value should be conditioned on the individual's inclusion in the accessible population, i.e. @p rho; the default is @c FALSE

Details

The @c E.vacc function complements the probability calculation @c g and computes the expected vaccination count that an individual would have received during a specified set of vaccination activities. The vaccination activities are described by inputs @p v and @p N; @p v provides the number of doses distributed and @p N provides the target population size for each activity. An individual's probability is determined by inputs @p z and @p w; @p z indicates which activities the individual is eligible for and @p w provides the pseudo-campaign weighting that specifies the probability of child receiving a routine vaccination each year. Finally, the scalar parameters, @p alpha and @p rho, quantify campaign efficiency and the size of accessible population, respectively.

Value

the expected number of vaccinations for an individual with eligibility vector @p z and weight vector @p w during the campaigns described by @p v and @p N

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

Examples

```
E.vacc( z = c(0,1), w = c(0.5,0.5), v = c(500,500), N = c(1000,1000), alpha = 0, rho = 0.9 )
```

---

g *Calculates the vaccination probability for individuals.*

Description

Calculates the vaccination probability for individuals.

Usage

```
g(z, w, v, N, alpha, rho, log=FALSE)
```

Arguments

z	the eligibility vector for the individual, i.e. @codez[j] = 1 if this individual is eligible for campaign @c j, otherwise @codez[j] = 0
w	the weight vector for the individual, i.e. @code w[k] gives the probability of the individual receiving the routine vaccination in year @c k
v	the doses vector providing the number of vaccine doses nominally distributed in each campaign

N	the population size vector specifying the number of people in the target age range for each campaign
alpha	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
rho	the proportion of the population (scalar) that can be vaccinated, i.e @code1 - rho is the @e inaccessible portion of the general population.
log	a flag indicating whether the log probability should be returned; the default is @c FALSE

Details

The @c g function is used to calculate the probability that an individual has been vaccinated during a set of vaccination activities. The vaccination activities are described by inputs @p v and @p N; @p v specifies the number of doses distributed and @p N specifies the target population size for each activity. An individual's probability is determined by inputs @p z and @p w; @p z indicates which activities the individual is eligible for and @p w provides the pseudo-campaign weighting that specifies the probability of child receiving a routine vaccination each year. Finally, the scalar parameters, @p alpha and @p rho, quantify campaign efficiency and the size of accessible population, respectively.

Value

the probability of vaccination for an individual with eligibility vector @p z and weight vector @p w during the campaigns described by @p v and @p N

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

Examples

```
g( z = c(0,1), w = c(0.5,0.5), v = c(500,500), N = c(1000,1000), alpha = 0, rho = 0.9 )
```

---

get.E.vacc *Calculates the expected number of vaccinations for all individuals...*

Description

Calculates the expected number of vaccinations for all individuals in a group of observations and under a described set of vaccination activities (e.g. campaigns).

Usage

```
get.E.vacc(obs, camps, trans.rho, alpha, z, w, cdf.fun,
    cond.on.accessible=FALSE, ...)
```

Arguments

obs	a data frame with one row per observation and columns for @c date of observation and @c age of individual on that date; Note: all ages should be in months
camps	a data frame with one row per activity and columns providing the activity's @c date, targeted age range (@c age.low and @c age.high), size of targeted population (@c N) and the number of vaccine doses nominally distributed (@c v)
trans.rho	the log-odds (aka logit) of @c rho, the proportion of the population (scalar) that can be vaccinated; see note above
alpha	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
z	the eligibility matrix for each individual/campaign pairing, i.e. @codez[i,j] = 1 if this individual @c i is eligible for campaign @c j, otherwise @codez[i,j] = 0; Note: if @c z is @c NULL, then the eligibilty matrix is calculated from @p obs and @p camps parameters using @code z.matrix function.
w	the weight matrix for each individual/campaign pairing, i.e. @code w[i,k] gives the probability of individual @c i receiving a routine vaccination in year @c k Note: if @c w is @c NULL, then the weight matrix is calculated from @p obs, @p camps, @p cdf.fun and @p ... parameters using @code w.matrix function.
cdf.fun	a cumulative distribution function (CDF) used to calculate the probability of routine vaccination based upon age, e.g. @code constant.post.9mo . Note: this parameter is only required if the weight matrix, @p w, is <u>not</u> provided.
cond.on.accessible	a flag indicating whether the return values should be conditioned on the individual's inclusion in the accessible population, i.e. @p rho; the default is @c FALSE
...	any additional arguments to the @p cdf.fun, e.g. @c lambda Note: this parameter is only required if the weight matrix, @p w, is not provided.

Details

The @c get.E.vacc function simply applies the @code E.vacc function to all the entries in observations data frame, @p obs. The @p obs argument should contain the immunization information as derived, for example, from a Demographic and Health Survey (DHS). The @c get.E.vacc function uses the observations and vaccination activity descriptions, @p camps, to calculate the eligibility matrix, @code z.matrix and the weight matrix @code w.matrix if not provided. Then the expected vaccination doses are calculated by calling @c E.vacc with each individual's eligibility and weight

vector as well as the relevant vaccination activity information (i.e. @p camps$N and @p camps$v), the efficiency parameter (@p alpha), the accessibility multiplier (@c rho) and conditionality flag (@p cond.on.accessible).

Value

a vector containing the expected number of vaccinations for each individual in observations data frame, @p obs.

@see E.vacc @see z.matrix @see w.matrix @see nine.month.pointmass @see constant.post.9mo

Note

The @p trans.rho argument represents the @linklogit of @c rho, i.e. @code trans.rho = log(rho / (1-rho)) , where @c rho is the proportion of population that is accessible to any vaccination activity and @code 1-rho the inaccessible portion. It follows then that @c rho is calculated as the inverse-logit (or logistic function) of @p trans.rho, that is: @code rho <- exp(trans.rho) / (1 + exp(trans.rho))

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

`get.prob.vacc` *Calculates the vaccination probability for all individuals in a group...*

Description

Calculates the vaccination probability for all individuals in a group of observations and under a described set of vaccination activities (e.g. campaigns).

Usage

```
get.prob.vacc(obs, camps, trans.rho, alpha, z, w, cdf.fun, ...)
```

Arguments

`obs`	a data frame with one row per observation and columns for @c date of observation and @c age of individual on that date; Note: all ages should be in months
`camps`	a data frame with one row per activity and columns providing the activity's @c date, targeted age range (@c age.low and @c age.high), size of targeted population (@c N) and the number of vaccine doses nominally distributed (@c v)

trans.rho	the log-odds (aka logit) of @c rho, the proportion of the population (scalar) that can be vaccinated; see note above
alpha	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
z	the eligibility matrix for each individual/campaign pairing, i.e. @codez[i,j] = 1 if this individual @c i is eligible for campaign @c j, otherwise @codez[i,j] = 0; Note: if @c z is @c NULL, then the eligibilty matrix is calculated from @p obs and @p camps parameters using @code z.matrix function.
w	the weight matrix for each individual/campaign pairing, i.e. @code w[i,k] gives the probability of individual @c i receiving a routine vaccination in year @c k Note: if @c w is @c NULL, then the weight matrix is calculated from @p obs, @p camps, @p cdf.fun and @p ... parameters using @code w.matrix function.
cdf.fun	a cumulative distribution function (CDF) used to calculate the probability of routine vaccination based upon age, e.g. @code constant.post.9mo . Note: this parameter is only required if the weight matrix, @p w, is <u>not</u> provided.
...	any additional arguments to the @p cdf.fun, e.g. @c lambda Note: this parameter is only required if the weight matrix, @p w, is not provided.

Details

The @c get.prob.vacc function simply applies the @code g function to all the entries in observations data frame, @p obs. The @p obs argument should contain the immunization information as derived, for example, from a Demographic and Health Survey (DHS). The @c get.prob.vacc function uses the observations and vaccination activity descriptions, @p camps, to calculate the eligibility matrix, @code z.matrix and the weight matrix @code w.matrix , if not provided. Then the vaccination probabilities are calculated by calling @c g with each individual's eligibility and weight vector as well as the relevant vaccination activity information (i.e. @p camps$N and @p camps$v), the efficiency parameter (@p alpha) and the accessibility multiplier (@c rho).

Value

a vector containing the probability of vaccination for each individual in observations data frame, @p obs.

@see g @see z.matrix @see w.matrix @see nine.month.pointmass @see constant.post.9mo

Note

The @p trans.rho argument represents the @linklogit of @c rho, i.e. @code trans.rho = log(rho / (1-rho)) , where @c rho is the proportion of population that is accessible to any vaccination activity and @code 1-rho the inaccessible portion. It follows then that @c rho is calculated as the inverse-logit (or logistic function) of @p trans.rho, that is: @code rho <- exp(trans.rho) / (1 + exp(trans.rho))

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

`ll.coverage` *Gets the log likelihood of a set of observations.*

Description

Gets the log likelihood of a set of observations.

Usage

```
ll.coverage(rho, alpha, obs, campaigns, z, w, cdf.fun, ...)
```

Arguments

`rho`	the rho parameter
`alpha`	the alpha parameter
`obs`	the observations. Must have columns y, date, age
`campaigns`	campaign definitions. Must have columns date, v, N, age.low, age.high
`z`	the z matrix. calculated if null
`w`	the w matrix. calculated if null
`cdf.fun`	the cdf function used to calculate the w.matrix, not needed if w is provided. This is large F in paper eq. 7,8
`...`	additional arguments to cdf.fun

Value

the log likelihood of the observations given the parameters

ll.coverage.optim *Log likelihood for using optim.*

Description

Log likelihood for using optim.

Usage

```
ll.coverage.optim(theta, obs, campaigns, z, cdf.fun, ...)
```

Arguments

theta	the parameter vector (trans.rho, alpha), where rho = exp(trans.rho)/(1+exp(trans.rho))
obs	the observations
campaigns	the campaigns
z	the z matrix
cdf.fun	the cdf function
...	additional arguments to cdf.fun

Details

Assumes weight matrix parameter is not being fit.

Value

the log likelihood

ll.coverage.optim.const.post.9m *Log likelihood for using optim.*

Description

Log likelihood for using optim.

Usage

```
ll.coverage.optim.const.post.9m(theta, obs, campaigns, z)
```

Arguments

theta	the parameter vector (trans.rho, alpha, lambda), where rho = exp(trans.rho)/(1+exp(trans.rho))
obs	the observations
campaigns	the campaigns
z	the z matrix

Details

Assumes that there there is a constant vaccination rate after 9 months of age

Value

the log likelihood

`logistic` *Defines a simple logistic function, @f$ e^x / (1 + e^x) @f$.*

Description

Defines a simple logistic function, @f$ e^x / (1 + e^x) @f$.

Usage

```
logistic(x)
```

Arguments

x	the variable to calculate the logistic function for

Details

The logistic function is used repeatedly in the coverage analysis so it defined here to improve the readability of those calculations.

Value

@code exp(x) / (1 + exp(x))

References

http://en.wikipedia.org/wiki/Logistic_function

Examples

```
x <- c( -10:-2, seq(-2,2,0.2), 2:10 )
plot( x=x, y=logistic(x) )
```

`make.sample.pop` *Generates a sample survey population with ages uniformly distributed...*

Description

Generates a sample survey population with ages uniformly distributed in the specified range, (@p age.low, @p age.high), and all assigned the provided observation @p date.

Usage

```
make.sample.pop(N, age.low, age.high, date)
```

Arguments

`N`	the number of samples, i.e. population size
`age.low`	the minimum age (in months) for the sample population
`age.high`	the maximum age (in months) for the sample population
`date`	the observation date to use for all samples

Value

a @c data.frame with @p N rows and two columns: @c date and @c age

@see runif

Note

The lower and upper age limits will not be generated. This function uses @coderunif to generate age values and @c runif does not generate either extreme unless @code low = high or @code max - min is small.

Note

If @p age.high < @p age.low, all generated ages will be @c NaN.

Examples

```
make.sample.pop( N=10, age.low=0, age.high=25, date="2012-01-01" )

N <- 10
dates <- c( "2005-05-01" ,"2007-07-02", "2009-09-03" )
make.sample.pop( N, 0, 25, array(dates,dim=N) )
```

`mcmc.converge` *Function to assess MCMC convergance of a chain given a matrix of MCMC...*

Description

Function to assess MCMC convergance of a chain given a matrix of MCMC iterations.

Usage

```
mcmc.converge(chains)
```

Arguments

`chains`	a matrix of n.interations X n.chains with results

Value

TBD

`mcmc.estimate` *Quick and dirty roll your own MCMC for determining the CI for rho...*

Description

Quick and dirty roll your own MCMC for determining the CI for rho and alpha.

Usage

```
mcmc.estimate(obs, campaigns, keep.from=1, n.iter=1000, n.chains=5,
    trans.rho.start=0.5, alpha.start=0, trans.rho.sd=0.1, alpha.sd=0.1,
    routine.cdf="9m.pointmass", routine.par.start=c(),
    routine.par.sd=rep(0.1, length(routine.par.start)), z,
    plot.as.go=FALSE, plot.freq=10)
```

Arguments

`obs`	the number of observations
`campaigns`	TBD
`keep.from`	TBD
`n.iter`	the number of interations to do
`n.chains`	the number of chains to run
`trans.rho.start`	starting point for rho

alpha.start	starting point for alpha
trans.rho.sd	standard deviation to use for proposal rhos
alpha.sd	standard deviation to us for proposal alphas
routine.cdf	TBD
routine.par.start	TBD
routine.par.sd	TBD
z	TBD
plot.as.go	TBD
plot.freq	TBD

Value

matrix of all of the chains for alpha and trans.rho.start

`nine.month.pointmass` *Defines a simple cumulative distribution function (CDF) with...*

Description

Defines a simple cumulative distribution function (CDF) with a point mass at nine (9) months.

Usage

```
nine.month.pointmass(age)
```

Arguments

age	the age of the individual

Details

The cumulative distribution function determines an individual's probabilty of receiving a routine vaccination. It is passed into the weighting calculation function (@codew.matrix) by various coverage analysis functions such as @code ll.coverage and @code mcmc.estimate . A user specifies, by name, which CDF to use when an analysis function is invoked, e.g. @code mcmc.estimate(..., routine.cdf = "9m.pointmass", ...) . See the documentation for the respective analysis functions for details about which CDFs are supported.

Value

zero (0) if `@p age < 9` and one (1) otherwise

@see w.matrix @see ll.coverage @see mcmc.estimate

Examples

```
x <- 0:24
plot( x=x, y=nine.month.pointmass(x) )
```

`sim.n.vacc.sample.pop` *Calculates the number of vaccinations each member of a sample population...*

Description

Calculates the number of vaccinations each member of a sample population would receive based upon probabilities generated from the campaigns descriptions and provided the @p rho and @p alpha values.

Usage

```
sim.n.vacc.sample.pop(obs, camps, rho, alpha, cond.on.accessible=FALSE,
    z, w, cdf.fun, ...)
```

Arguments

`obs`	the synthetic population, i.e. a data frame with one row per observation and columns for @c date of observation and @c age of individual on that date; Note: all ages should be in months
`camps`	a data frame with one row per activity and columns providing the activity's @c date, targeted age range (@c age.low and @c age.high), size of targeted population (@c N) and the number of vaccine doses nominally distributed (@c v)
`rho`	the proportion of the population (scalar) that can be vaccinated, i.e @code1 - rho is the @e inaccessible portion of the general population.
`alpha`	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
`cond.on.accessible`	a flag indicating whether the probabilities should be conditioned on the individual's inclusion in the accessible population, i.e. @p rho; the default is @c FALSE
`z`	the eligibility matrix for each individual/campaign pairing, i.e. @codez[i,j] = 1 if this individual @c i is eligible for campaign @c j, otherwise @codez[i,j] = 0; Note: if @c z is @c NULL, then the eligibilty matrix is calculated from @p obs and @p camps parameters using @code z.matrix function.
`w`	the weight matrix for each individual/campaign pairing, i.e. @code w[i,k] gives the probability of individual @c i receiving a routine vaccination in year @c k Note: if @c w is @c NULL, then the weight matrix is calculated from @p obs, @p camps, @p cdf.fun and @p ... parameters using @code w.matrix function.
`cdf.fun`	a cumulative distribution function (CDF) used to calculate the probability of routine vaccination based upon age, e.g. @code constant.post.9mo . Note: this parameter is only required if the weight matrix, @p w, is <u>not</u> provided.

... any additional arguments to the @p cdf.fun, e.g. @c lambda Note: this parameter is only required if the weight matrix, @p w, is not provided.

Details

TBD: Add details

Value

a vector containing the number of vaccinations each individual in observation set, @p obs, would have received

@see z.matrix @see w.matrix @see nine.month.pointmass @see constant.post.9mo @see get.win.smooth.expected.mcmcres @see get.win.smooth.thresh.mcmcres

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

Examples

```
N <- 100
dates <- as.Date( c( "2005-05-01", "2007-07-02", "2009-09-03" ) )
sample.obs   <- make.sample.pop( N=N, age.low=0, age.high=25,
date=array(dates,dim=N) )
sample.camps <- data.frame( date=(dates - 3*DAYS.PER.MONTH),
N=(0.5*N), v=(0.5*N),
age.low=8, age.high=20, is.SIA=1 )

sim.n.vacc.sample.pop( sample.obs, sample.camps, rho=0.9, alpha=0,
cdf.fun=nine.month.pointmass )
```

---

vaccinate.sample.pop *Vaccinates a sample population based on probabilities generated from the...*

Description

Vaccinates a sample population based on probabilities generated from the campaigns descriptions and the @p rho and @p alpha values.

Usage

```
vaccinate.sample.pop(obs, camps, rho, alpha, cdf.fun, ...)
```

Arguments

obs	the synthetic population, i.e. a data frame with one row per observation and columns for @c date of observation and @c age of individual on that date; Note: all ages should be in months
camps	a data frame with one row per activity and columns providing the activity's @c date, targeted age range (@c age.low and @c age.high), size of targeted population (@c N) and the number of vaccine doses nominally distributed (@c v)
rho	the proportion of the population (scalar) that can be vaccinated, i.e @code1 - rho is the @e inaccessible portion of the general population.
alpha	the vaccination efficiency parameter (scalar) for the campaign; @codealpha = -inf implies perfect efficiency, @codealpha = 0 implies random efficiency; see note above
cdf.fun	a cumulative distribution function (CDF) used to calculate the probability of routine vaccination based upon age, e.g. @code constant.post.9mo .
...	any additional arguments to the @p cdf.fun, e.g. @c lambda

Details

TBD: Add details

Value

a copy of @p obs data frame with two new columns appended: @c y (vaccination status, i.e. @c 0 or @c 1) and @c p.vac (vaccination probability)

@see g @see z.matrix @see w.matrix @see nine.month.pointmass @see constant.post.9mo

Note

The inefficiency measure @p alpha is defined as the natural log of psi, i.e. @codepsi = e^alpha. @codepsi = 0 (@codealpha = -inf) represents perfect efficiency, i.e. when every dose results in an additional vaccinee. @codepsi = 1 (@codealpha = 0) represents random efficiency, i.e. when probability of any dose resulting in an additional vaccinee is independent of receiving a dose previously during the same campaign.

Examples

```
N <- 100
dates <- as.Date( c( "2005-05-01", "2007-07-02", "2009-09-03" ) )
sample.obs   <- make.sample.pop( N=N, age.low=0, age.high=25,
date=array(dates,dim=N) )
sample.camps <- data.frame( date=(dates - 3*DAYS.PER.MONTH),
N=(0.5*N), v=(0.5*N),
age.low=8, age.high=20, is.SIA=1 )

vaccinate.sample.pop( sample.obs, sample.camps, rho=0.9, alpha=0,
cdf.fun=nine.month.pointmass )
```

`w.matrix` *Creates a weight matrix from (1) immunization information for a set...*

Description

Creates a weight matrix from (1) immunization information for a set of individuals, e.g. observations derived from a Demographic and Health Survey (DHS), (2) a set of vaccination activity descriptions, and (3) a cumulative distribution function (CDF).

Usage

```
w.matrix(obs, activities, cdf.fun, ...)
```

Arguments

`obs`	a data frame with one row per observation and columns for date of observation and age of individual on that date; Note: all ages should be in months
`activities`	a data frame with one row per activity and columns providing the activity's date and the @c is.SIA flag indicating whether the activity is a Supplemental Immunization Activity (SIA) or routine vaccination.
`cdf.fun`	a cumulative distribution function (CDF) used to calculate the probability of routine vaccination during each activity as a function of age, e.g. @code constant.post.9mo .
`...`	additional arguments to the @p cdf.fun, e.g. @c lambda

Details

The @c w.matrix function loops through all the provided activities and determines weights for all observations. These weights give the probability of an individual having the "opportunity" to be vaccinated during a year's routine activities.

Value

a matrix with one row per observation and one column per activity; matrix cells contain an "opportunity" weight which is 0 for SIA campaigns and calculated as using the @p cdf.fun for routine vaccination activities

@see nine.month.pointmass @see constant.post.9mo

Note

For all Supplemental Immunization Activities (SIAs), i.e. campaigns, the routine vaccination opportunity weight defaults to zero (0).

For each routine vaccination activity, the opportunity weight is based on the individual's age at the start of the activity and the length of exposure to that year's activity (12 months most years, but truncated in the survey year). Specifically, the weight is calculated as: @f[cdf.fun(xij + lj) -

cdf.fun(xij) @f] where @c xij is the age of individual @c i at the start of year @c j and @c lj is the length of exposure to routine vaccination in year @c j.

For length of the exposure to the routine vaccination, @c lj, there are three possible cases: @li a. if survey date is before activity year, then the individual is not "exposed" to this routine vaccination and hence @f$ lj = 0 months @f$ @li b. if survey date is during activity year, then the individual is "exposed" only up until the observation date and hence @f$ lj = (obs$date - activities$date) @f$ @li c. if survey date is after activity year, then the individual is "exposed" to this routine vaccination for the entire year and hence @f$ lj == 12 months @f$

Note

For routine vaccinations, the date provided (i.e. @p activities$date) is used as start of the activity year and hence is typically given as January 1st of that year, e.g. @c 2012-01-01.

`z.matrix` *Creates an eligibility matrix, i...*

Description

Creates an eligibility matrix, i.e. a @c z matrix, from immunization information for a set of individuals, e.g. observations derived from a Demographic and Health Survey (DHS), and a set of vaccination activity descriptions (e.g. campaigns).

Usage

```
z.matrix(obs, campaigns)
```

Arguments

`obs`	a data frame with one row per observation and columns for @c date of observation and @c age of individual on that date; Note: all ages should be in months
`campaigns`	a data frame with one row per activity and columns providing the activity's @c date and targeted age range, i.e. @c age.low and @c age.high

Details

The @c z.matrix function loops through all the provided observations and campaigns. For each pairing, it is first determined if the campaign occurred before or after the observation (survey) date. If the vaccination campaign was after the observation date, then the individual's eligibility status is set to zero (0). Otherwise, if the campaign was completed before survey, then the eligibility status is based on age. If the individual's age, at the time of the campaign, was within age range, @code[age.low, age.high], of the immunization activity's target population, then eligibility is set to one (1). If the individual's age was not within the target age range, then eligibility is zero (0).

Value

a matrix with one row per observation and one column per campaign; matrix cells contain a @c 1 if that individual was eligible for the campaign and @c 0 otherwise

Note

The @c z.matrix function only considers campaigns, i.e. Supplemental Immunization Activities (SIAs). For non-SIA immunization activities, e.g. routine vaccinations, the eligibility status is always set to zero (0).

Note

All vaccination activities are assumed to have been completed on the date provided (i.e. @p campaigns$date); separate start and end dates are not supported.

Index