Package 'SIRmcmc'

January 20, 2025

Type Package

Title Compartmental Susceptible-Infectious-Recovered (SIR) Model of Community and Household Infection

Version 1.1.1

Date 2023-12-22

Author F Scott Dahlgren and Ivo M Foppa

Maintainer F Scott Dahlgren <fdahlgr@gmail.com>

Description We build an Susceptible-Infectious-Recovered (SIR) model where the rate of infection is the sum of the household rate and the community rate. We estimate the posterior distribution of the parameters using the Metropolis algorithm. Further details may be found in: F Scott Dahlgren, Ivo M Foppa, Melissa S Stockwell, Celibell Y Vargas, Philip LaRussa, Carrie Reed (2021) ``Household transmission of influenza A and B within a prospective cohort during the 2013-2014 and 2014-2015 seasons" <doi:10.1002/sim.9181>.

License GPL (>= 2)

Imports Rcpp (>= 0.12.15), methods

LinkingTo Rcpp

NeedsCompilation yes

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-01-11 17:50:09 UTC

Contents

`community_attack_rate` *Compute the community attack rate*

Description

Computes the community attack rate for a a cohort using the value of a call to `household_transmission`.

Usage

```
community_attack_rate(SIRmcmc,probs=c(0.5,0.025,0.975))
```

Arguments

`SIRmcmc`	The value of a call to `household_transmission`.
`probs`	A numeric vector of the quantiles of the posterior distribution. The default is the median and the central 95% credible region.

Details

Computes the posterior probability distribution of the community attack rate from the Metropolis algoritm. Returns quantiles of the distribution specified in the `probs` argument.

Value

An array of community attack rates. The first dimension is the value of epsilon in `SIRmcmc`. The second dimension is the posterior probability in the `probs` argument

---

`hh.transmission` *Simulated data under the SIR.*

Description

Simulated data under the SIR model, without covariates.

Usage

```
data("hh.transmission")
```

Format

A data frame with 1000 observations on the following 3 variables.

`household` A numeric vector of household IDs.

`onset` Date of onset for cases

`transmission` A factor with levels `Community` `Household` `None` representing source of infection.

hh.transmission.epsilon

Simulated data under SIR, with covariates.

Description

Simulated data under the SIR model with a high risk and low risk group. The first 1000 observations are identical to the data `hh.transmission`.

Usage

```
data("hh.transmission.epsilon")
```

Format

A data frame with 2000 observations on the following 4 variables.

`household` A numeric vector of household IDs.

`onset` Date of onset for cases.

`transmission` A factor with levels `Community` `Household` `None` representing source of infection.

`epsilon` A numeric vector of the value of epsilon used to simulate the data.

household_transmission

Estimate parameters from SIR model

Description

Use the Metropolis algorithm to estimate parameters from the SIR compartmental model

Usage

```
household_transmission(onset_date, household_index, covariate = NULL,
                       followup_time, iterations,
                       delta = c(0.1,0.3,0.4,0.1), plot_chain = TRUE,
                       index = 1, start_date = NULL, prior = unifprior,
                       constant_hazard = FALSE,...)
```

Arguments

onset_date	A vector of onset dates. If onset_date is a character vector, then it will be coerced to dates using as.Date. NA values are interpretted as remaining susceptible at the end of followup.
household_index	A vector identifying households, which should be the same length as the onset_date argument.
covariate	A vector identifying covariate patterns. If given, then it is interpretted as a factor. A value for epsilon will be given for each level of covariate. If NULL, then epsilon is not estimated.
followup_time	An integer for the followup time in days.
iterations	An integer for the number of iterations of the Metropolis algorithm.
delta	A vector of length 4 for tuning the acceptance rate of the Metropolis algorithm. The order is (1) alpha, (2) beta, (3) gamma, and (4) epsilon.
plot_chain	A boolean of whether to plot the value of the chain vs the iterate.
index	An integer for the index date. Probabilities are conditional on the index date. Any coordinates of onset_date equal to index will have a likelihood of 1. If you want unconditional probabilities, then index should be less than start_date.
start_date	Should be the same format as onset_date. Specifies the start of the followup period.
prior	A function to compute the prior probability of alpha, beta, gamma, and epsilon. Any user written function must take the arguments alpha, beta, gamma, epsilon, and esteps. Builtin functions are unifprior and prior. unifprior is uniform on (0.01,1000).
constant_hazard	If FALSE, then the algorithm computes a time dependent hazard for the cohort and the hazard from the community is proportional to the hazard in the cohort. If TRUE, then the algorithm assumes a constant hazard from the community.
...	Arguments to be passed to the function in the prior argument.

Details

If no covariates are supplied, only the model parameters alpha, beta, and gamma are estimated using a stepwise Metropolis algorithm. The model parameters are drawn from a uniform distribution on (0.1,1). The first step proposes a new alpha using rnorm and the mixing parameter delta[1]. The second and third steps are similar for beta and gamma. If covariates are supplied, the additional parameters, collectively called epsilon, are also estimated.

The algorithm assumes followup begins on start_date and lasts for followup_time days. Any coordinate of onset_date equal to index does not contribute to the likelihood.

Two priors are builtin: prior and unifprior. User defined prior functions must take the arguments alpha, beta, gamma, epslion, and esteps.

Value

An object of the class SIRmcmc.

See Also

prior unifprior

Examples

```
##A trivial example----------------------------------------------------------
library(graphics)
onset<-sample(c(seq(1,10),rep(Inf,20)),size=500,replace=TRUE)
hh<-sample(seq(1,300),size=500,replace=TRUE)
chain<-household_transmission(onset_date = onset, household_index = hh,
                              followup_time = 10, iterations = 100)
community_attack_rate(SIRmcmc=chain)
secondary_attack_rate(household_size=3,SIRmcmc=chain)

##An example with household transmission---------------------------------
library(graphics)
data(hh.transmission)
set.seed(1)
iterations<-100
T<-30
delta<-c(0.1,0.6,0.8)
index<-0
##Find the MCMC estimates of alpha, beta, and gamma
chain<-household_transmission(
    onset_date=hh.transmission$onset,
    household_index=hh.transmission$household,
    covariate=NULL,
    followup_time=T,
    iterations=iterations,
    delta=delta,
    prior=unifprior,
    index=index
)
#Tabulate true type of transmission
hh_table<-table(
    table(
        is.finite(MCMC_date(hh.transmission$onset)),
        hh.transmission$household)["TRUE",]
)
##Calculate the true SAR
truth_table<-table(hh.transmission$transmission)
truth<-unname(truth_table["Household"]/sum(hh_table[2:3]))
cat("\n\nTrue Value of SAR\n\n")
print(truth)
##Find point and 95% central creditable intervals for MCMC SAR
cat("\n\nMCMC Estimate of SAR\n\n")
secondary_attack_rate(household_size=2,SIRmcmc=chain)
days<-NULL
for(d in c(seq(1:5))){
    days<-c(days,as.character(d))
   a<-sum(table(tapply(X=hh.transmission$onset,INDEX=hh.transmission$household,FUN=diff))[days])
    cat(
```

```
        paste0(
            "\n\n",
            d,
            " Day Counting Estimate of SAR\n\n"
        )
    )
    ##Find point and 95% confidence intervals for normal approx to SAR
    print(
        a/sum(hh_table[2:3])+c(p=0,LB=-1,UB=1) *
        qnorm(p=0.975) *
        sqrt(a*(hh_table[2]+hh_table[3]-a)/(hh_table[2]+hh_table[3])^3)
    )
}


##An example with rate ratios----------------------------------------
## Not run:
    library(graphics)
    data(hh.transmission.epsilon)
    set.seed(1)
    iterations<-100
    T<-30
    delta<-c(0.1,0.1,0.1,0.1)
    index<-0
    ##Find the MCMC estimates of alpha, beta, gamma, and epsilon
    chain<-household_transmission(
        onset_date=hh.transmission.epsilon$onset,
        household_index=hh.transmission.epsilon$household,
        covariate=hh.transmission.epsilon$epsilon,
        followup_time=T,
        iterations=iterations,
        delta=delta,
        prior=unifprior,
        index=index
    )

    ##Find point and 95% central creditable intervals for MCMC SAR
    cat("\n\nMCMC Estimate of SAR\n\n")
    print(secondary_attack_rate(household_size=2,SIRmcmc=chain))
    for(e in c(1,5)){
        hh_table<-table(table(
            is.finite(MCMC_date(hh.transmission.epsilon$onset)),
            hh.transmission.epsilon$household,
            hh.transmission.epsilon$epsilon)["TRUE",,as.character(e)])
        ##Tabulate true type of transmission
        truth_table<-table(
            hh.transmission.epsilon$transmission[which(
                hh.transmission.epsilon$epsilon==e
            )])
        ##Calculate the true SAR
        truth<-unname(truth_table["Household"]/sum(hh_table[2:3]))
        cat("\n\nTrue Value of SAR\n\n")
        print(truth)
```

```
        days<-NULL
        for(d in c(seq(1:5))){
            days<-c(days,as.character(d))
            a<-sum(table(tapply(
             X=hh.transmission.epsilon$onset[which(hh.transmission.epsilon$epsilon==e)],
           INDEX=hh.transmission.epsilon$household[which(hh.transmission.epsilon$epsilon==e)],
                FUN=diff))[days])
            ##Find point and 95% confidence intervals for normal approx to SAR
            cat(paste0(
                "\n\n",
                d,
                " Day Counting Estimate of SAR\n\n"
            ))
            print(
                a/sum(hh_table[2:3])+c(p=0,LB=-1,UB=1)
                * qnorm(p=0.975)
                * sqrt(a*(hh_table[2]+hh_table[3]-a)/(hh_table[2]+hh_table[3])^3)
            )
        }
    }

## End(Not run)
```

`MCMC_date` *Convert dates to a list of extendend natural numbers.*

Description

Converts dates to a list of numbers representing the number of days from the start of followup until the start of the infectious period.

Usage

```
MCMC_date(dates,start_date=NULL)
```

Arguments

`dates`	A list of dates. May be of character, numeric, or date class.
`start_date`	The start date of follow up

Details

Covrerts dates to days of followup until the start of the infectious period. Missing data are set to infinity and are assumed susceptible during followup.

Value

A list of extended natural numbers.

prior *Compute prior probability of parameters*

Description

Compute the prior probability of alpha, beta, gamma, and epsilon

Usage

```
prior(alpha, beta, gamma, epsilon, esteps, params = list(alpha =
      list(location = 0, scale = 2), beta = list(shape = 0.01, rate = 0.01),
      gamma = list(shape = 0.01, rate = 0.01),
      epsilon = list(location = 0, scale = 2)))
```

Arguments

alpha	A number. The logarithm of alpha follows a logistic distribution.
beta	A number. beta follows a gammadistribution.
gamma	A number. gamma follows a gamma distribution.
epsilon	A number. The logarithm of epsilon follows a logistic distribution.
esteps	A number in (0,1). Used for logic as to whether to compute the prior probability of epsilon (1) or not (0).
params	A list of parameters for the prior distributions.

Value

A probability of the model parameters under the prior distributions.

See Also

dlogis dgamma household_transmission

secondary_attack_rate *Compute the secondary attack rate*

Description

Using the value of a call to household_transmission, computes the secondary attack rate for households.

Usage

```
secondary_attack_rate(household_size,SIRmcmc,probs=c(0.5,0.025,0.975))
```

Arguments

household_size	A numeric vector containing the number of people in a household.
SIRmcmc	The value of a call to household_transmission.
probs	A numeric vector of the quantiles of the posterior distribution. The default is the median and the central 95% credible region.

Details

Computes the posterior probability distribution of the secondary attack rate from the Metropolis algorithm. Returns quantiles of the distribution specified in the probs argument.

Value

An array with secondary attack rates.

The first dimension of the array is the household size.

The second dimension of the array is the quantiles of the posterior distribution in the probs argument.

The third dimension of the array is the value of epsilon in SIRmcmc.

---

unifprior *A uniform prior on the model parameters*

Description

Compute a uniform prior on alpha, beta, gamma, and epsilon

Usage

```
unifprior(alpha, beta, gamma, epsilon, esteps, UB = 1000, LB = 0.01)
```

Arguments

alpha	A number compared to the upper bound UB and the lower bound LB.
beta	A number compared to the upper bound UB and the lower bound LB.
gamma	A number compared to the upper bound UB and the lower bound LB.
epsilon	A number compared to the upper bound UB and the lower bound LB.
esteps	A number in (0,1). Used for logic as to whether to compute the prior probability of epsilon (1) or not (0).
UB	A number used as an upper bound on the model parameters.
LB	A number used as a lower bound on the model parameters.

Value

If all the model parameters are between the lower bound and the upper bound, then unifprior returns 1. Otherwise, unifprior returns 0.

See Also

dlogis dgamma household_transmission prior

Index