Package 'hte'

June 13, 2023

R topics documented:

`hte-package` *Hospital transmission estimation*

Description

The **hte** package ...

Author(s)

Aaron A. King

`Bernoulli_filter` *Bernoulli filter*

Description

Filter the individual patient data with given forces of infection, recovery rates, and test characteristics.

Usage

```
Bernoulli_filter(data, lambda, gamma, theta)

Bfilter(data, theta)
```

Arguments

`data`	data set
`lambda`	unit-specific force of infection
`gamma`	unit-specific recovery rate
`theta`	list of parameters

Details

`Bernoulli_filter` runs a Bernoulli filter, updating the expected prevalence. It returns the log likelihood, occupancy, and expected prevalences.

`Bfilter` uses an alternative algorithm.

Value

`Bernoulli_filter` returns a `tibble` containing the expected prevalences (for both isolated and un-isolated patients), unit occupancy, force of infection, and conditional log likelihood for each unit at each event time. The sum of the log likelihood column (`logLik`) is the log likelihood of the data.

`Bfilter` returns the log likelihood of the furnished data.

Examples

```
library(tidyverse)

set.seed(626292345)

fake_data |>
  arrange(patient,time) -> dat

theta <- list(
  lambda=c(out=0,A=0.01,B=0.1,C=0.2,D=0.3,E=0.5),
  gamma=c(out=0.1,A=0.1,B=0.1,C=0.1,D=0.1,E=0.1),
```

```
    p0=0.2,
    isol_factor=0.1,
    alpha=0.05,
    beta=0.2
  )

  dat |>
    Bfilter(theta) -> ll1
  sum(ll1)

  dat |>
    Bernoulli_filter(
      lambda=theta$lambda,
      gamma=theta$gamma,
      theta
    ) -> f
  f |> filter(logLik!=0) |> pull(logLik) -> ll2
  sum(ll2)

  f

  f |>
    select(unit,time,prev_i,prev_u) |>
    pivot_longer(c(prev_i,prev_u)) |>
    group_by(unit) |>
    ## prevalence is not estimated outside the hospital
    filter(!all(is.na(value))) |>
    ungroup() |>
    ggplot(aes(x=time,color=name,y=value))+
    geom_line(alpha=0.5)+
    scale_color_manual(values=c(prev_i="blue",prev_u="red"))+
    facet_grid(unit~.,labeller=label_both)+
    labs(y="prevalence")+
    theme_bw()
```

`coal_last` *Coalesce with last*

Description

Fills NA with previous non-NA.

Usage

```
coal_last(x)
```

Arguments

x	vector

fake_data *Fake hospital movement, testing, and isolation data*

Description

A simulated outbreak

Details

Data are generated using the default settings of simuldat.

The data were generated by:

```
set.seed(339613584)
simuldat(verbose=TRUE) -> fake_data
save(fake_data,file="fake_data.rda",compress="xz")
```

See Also

More on simulated data: simuldat()

Examples

```
library(tidyverse)
library(lubridate)

## Examine the data:
fake_data

## Verify certain conditions hold:
stopifnot(
  `admission condition violation`=fake_data |>
    group_by(patient,visit) |> slice_head() |>
    filter(event!="admit") |> nrow()==0,
  `discharge condition violation`=fake_data |>
    group_by(patient,visit) |> slice_tail() |>
    filter(event!="discharge",event!="stop") |> nrow()==0,
  `unit violation`=fake_data |> filter(is.na(unit)) |> nrow()==0,
  `event violation`=fake_data |> filter(is.na(event)) |> nrow()==0
)

fake_data |>
  mutate(
    time=as.numeric(
      as.duration(
        interval(date,start="2000-01-01T00:00:00+0000")
      ),
      units="day"
    )
  ) -> fake_data
```

```
fake_data |>
  group_by(patient,visit) |>
  summarize(dur=max(time)-min(time)) |>
  ungroup() |>
  group_by(patient) |>
  summarize(dur=sum(dur)) |>
  ggplot(aes(x=log10(dur)))+
  geom_histogram(bins=20)+
  labs(title="total duration of hospitalization")+
  theme_bw()

fake_data |>
  group_by(patient,visit) |>
  summarize(dur=max(time)-min(time)) |>
  ungroup() |>
  ggplot(aes(x=log10(dur)))+
  geom_histogram(bins=40)+
  labs(title="duration of hospital visit")+
  theme_bw()

fake_data |>
  filter(
    event!="test",
    event!="isolate",
    event!="release",
    event!="stop"
  ) |>
  group_by(patient,visit) |>
  arrange(time) |>
  mutate(dur=lead(time)-time) |>
  ungroup() |>
  filter(unit!="out",!is.na(dur)) |>
  ggplot(aes(x=log10(dur),fill=unit,group=unit))+
  geom_histogram(aes(y=after_stat(density)),bins=40)+
  facet_grid(unit~.,scales="free_y")+
  labs(title="duration of stay by unit")+
  theme_bw()

fake_data |>
  group_by(patient) |>
  summarize(ntest=sum(event=="test")) |>
  ungroup() |>
  ggplot(aes(x=ntest))+
  geom_histogram(binwidth=1,center=0)+
  labs(title="number of tests per patient")+
  theme_bw()

fake_data |>
  arrange(time) |>
  mutate(
    dn=case_when(
      event=="admit"~1L,
```

```
      event=="discharge"~-1L,
      TRUE~0L
    ),
    occ=cumsum(dn)
  ) |>
  ggplot(aes(x=date,y=occ))+
  geom_step()+
  labs(title="hospital occupancy")+
  theme_bw()

fake_data |>
  arrange(time) |>
  select(date,test.result=result,isol,infected) |>
  pivot_longer(c(test.result,isol,infected)) |>
  filter(!is.na(value)) |>
  ggplot(aes(x=date,y=value,color=name))+
  geom_point()+
  geom_smooth()+
  guides(color="none")+
  labs(
    title="infection and isolation status, test results",
    y=""
  )+
  facet_grid(name~.)+
  theme_bw()

fake_data |>
  filter(event=="test") |>
  mutate(
    interval=cut(time,breaks=72,ordered=TRUE)
  ) |>
  select(interval,time,infected,isol,result) |>
  pivot_longer(c(infected,isol,result)) |>
  group_by(name,interval) |>
  summarize(
    time=mean(time),
    prev=mean(value),
    n=n()
  ) |>
  ungroup() |>
  ggplot(aes(x=time,y=prev,group=name,fill=name))+
  geom_col(position="dodge")+
  labs(title="infection, isolation, and detection through time")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90))

fake_data |>
  filter(event=="test") |>
  select(infected,result) |>
  count(infected,result) |>
  group_by(infected) |>
  mutate(prob=n/sum(n)) |>
  ungroup()
```

`independent` *Independent infection model*

Description

Model under which each patient's trajectory of infection is independent, conditional on the force of infection.

Usage

```
indep_homog_filter(params, data)

indep_homog_objfun(params, data, est = character(0))

indep_unit_spec_filter(params, data)

indep_unit_spec_objfun(params, data, est = character(0))
```

Arguments

`params`	named vector of parameters
`data`	patient movement, isolation, and testing data
`est`	names of parameters to estimate

Details

Parameters in the model include:

lambda force of infection. In the homogeneous model, this is constant across units in the hospital; in the unit-specific model, there is one value of lambda for each unit.

lambda.out force of infection outside the hospital

gamma recovery rate

alpha,beta false positive and negative probabilities

p0 probability of infection on admission

isol_factor multiplicative effect of contact isolation on susceptibility

`indep_homog_filter` runs a Bernoulli filter for the independent model with a global lambda and gamma.

`indep_homog_objfun` is a stateful objective function for the independent model with a global lambda and gamma.

`indep_unit_spec_filter` runs a Bernoulli filter for the independent model with a unit-specific lambda.

`indep_unit_spec_objfun` is a stateful objective function for the independent model with unit-specific lambda.

See Also

More on the independent infection model: simuldat()

More on stateful objective functions: stobfun(), transmission

Examples

```
library(tidyverse)
library(lubridate)
library(pomp)
library(hte)

set.seed(626292345)

## Select some data:
fake_data |>
  filter(
    event!="test" | time < 250,
    time < 1000
  ) |>
  arrange(patient,time) -> dat

## Create an objective function:
indep_homog_objfun(
  params=c(
    lambda.out=0.05,lambda=0.1,gamma=0.01,p0=0.1,
    isol_factor=0.2,alpha=0.02,beta=0.1
  ),
  est=c("lambda","gamma"),
  data=dat
) -> f

## Fit the model:
optim(
  par=log(c(0.1,0.01)),
  fn=f,
  control=list(reltol=1e-3)
) -> out
f(out$par)
coef(f)

## Examine the filter results:
indep_homog_filter(params=coef(f),data=dat) -> ff
ff |> filter(logLik!=0)

## Construct a slice:
indep_homog_objfun(
  params=coef(f),
  est=c("lambda"),
  data=dat
) -> g

log.lambda <- seq(log(0.001),to=log(1),length=10)
```

```
plot(exp(log.lambda),sapply(log.lambda,g),log='x')

## Construct an objective function for the unit-specific model:
theta <- coef(f)
indep_unit_spec_objfun(
  params=c(
    lambda=setNames(rep.int(theta["lambda"],6),unique(dat$unit)),
    theta[c("p0","gamma","alpha","beta","isol_factor")]
  ),
  est=c(
    "lambda.A","lambda.B","lambda.C","lambda.D","lambda.E","lambda.out","gamma"
  ),
  data=dat
) -> h

## Fit the model:
optim(
  par=log(coef(h)[
    c("lambda.A","lambda.B","lambda.C","lambda.D","lambda.E","lambda.out","gamma")
  ]),
  fn=h,
  control=list(reltol=1e-3)
) -> out
h(out$par)
coef(h)
```

---

simuldat *simuldat*

Description

simuldat simulates data representing the flow of a body of patients through a hospital over a specified window of time.

simul_patient simulates a single patient's history of movement, testing, isolation, and infection.

Usage

```
simuldat(
  nbeds = c(50, 60),
  arrival = 20,
  window = c("1999-12-31T23:59:59+0000", "2003-01-01T00:00:00+0000"),
 units = list(A = list(shape = 10, scale = 2/10), B = list(shape = 5, scale = 0.5/5), C
   = list(shape = 1, scale = 0.5/1), D = list(shape = 1, scale = 3/1), E = list(shape =
    0.2, scale = 8/0.2), out = list(shape = 0.5, scale = 300/0.5)),
  visits = list(size = 0.5, mu = 9),
  uperv = list(size = 1, mu = 0.5),
  min_dur = 1/24,
  testing_freq = c(A = NA, B = NA, C = 1/7, D = 1/7, E = 1/7, out = NA),
  isolation = list(on = 1/50, off = 1/50),
```

```
  infection = list(lambda = c(A = 0.01, B = 0.02, C = 0.001, D = 0.5, E = 0.1, out =
    0.05), gamma = 0.01, p0 = 0.1, isol_factor = 0.2),
  alpha = 0.02,
  beta = 0.1,
  verbose = getOption("verbose", TRUE)
)

simul_patient(
  patient,
  t0,
  tf,
  units,
  visits,
  uperv,
  min_dur,
  testing_freq,
  isolation,
  infection,
  alpha,
  beta
)

indep_infect(lambda, gamma, p0, isol_factor, times, loc, isol)
```

Arguments

nbeds	upper and lower bounds on number of beds
arrival	Poisson arrival rate of new patients
window	window of simulation
units	a named list with one entry per unit. Each entry is itself a list with the parameters of the Gamma-distribution for the duration of stay in the unit.
visits	list containing parameters of a negative binomial distribution for the number of visits per patient.
uperv	list containing parameters for a negative binomial distribution for the number of units visited per visit.
min_dur	minimum duration of stay in any unit
testing_freq	named numeric vector of unit-specific testing frequencies.
isolation	list containing parameters of the isolation model
infection	parameters of the infection model
alpha, beta	false positive and negative testing error rates
verbose	run-time information?
patient	patient name or number
t0, tf	initial and final times of patient itinerary
lambda	force of infection

`gamma`	recovery rate
`p0`	initial probability of infection
`isol_factor`	reduction in susceptibility due to isolation
`times`	times at which status is reported
`loc`	location of patient at each time
`isol`	isolation status

Details

Simulate hospital movement, testing, and isolation data.

Value

infection status vector

See Also

More on simulated data: `fake_data`

More on the independent infection model: `independent`

Examples

```
library(tidyverse)
library(lubridate)

## Examine the data:
fake_data

## Verify certain conditions hold:
stopifnot(
  `admission condition violation`=fake_data |>
    group_by(patient,visit) |> slice_head() |>
    filter(event!="admit") |> nrow()==0,
  `discharge condition violation`=fake_data |>
    group_by(patient,visit) |> slice_tail() |>
    filter(event!="discharge",event!="stop") |> nrow()==0,
  `unit violation`=fake_data |> filter(is.na(unit)) |> nrow()==0,
  `event violation`=fake_data |> filter(is.na(event)) |> nrow()==0
)

fake_data |>
  mutate(
    time=as.numeric(
      as.duration(
        interval(date,start="2000-01-01T00:00:00+0000")
      ),
      units="day"
    )
  ) -> fake_data
```

```
fake_data |>
  group_by(patient,visit) |>
  summarize(dur=max(time)-min(time)) |>
  ungroup() |>
  group_by(patient) |>
  summarize(dur=sum(dur)) |>
  ggplot(aes(x=log10(dur)))+
  geom_histogram(bins=20)+
  labs(title="total duration of hospitalization")+
  theme_bw()

fake_data |>
  group_by(patient,visit) |>
  summarize(dur=max(time)-min(time)) |>
  ungroup() |>
  ggplot(aes(x=log10(dur)))+
  geom_histogram(bins=40)+
  labs(title="duration of hospital visit")+
  theme_bw()

fake_data |>
  filter(
    event!="test",
    event!="isolate",
    event!="release",
    event!="stop"
  ) |>
  group_by(patient,visit) |>
  arrange(time) |>
  mutate(dur=lead(time)-time) |>
  ungroup() |>
  filter(unit!="out",!is.na(dur)) |>
  ggplot(aes(x=log10(dur),fill=unit,group=unit))+
  geom_histogram(aes(y=after_stat(density)),bins=40)+
  facet_grid(unit~.,scales="free_y")+
  labs(title="duration of stay by unit")+
  theme_bw()

fake_data |>
  group_by(patient) |>
  summarize(ntest=sum(event=="test")) |>
  ungroup() |>
  ggplot(aes(x=ntest))+
  geom_histogram(binwidth=1,center=0)+
  labs(title="number of tests per patient")+
  theme_bw()

fake_data |>
  arrange(time) |>
  mutate(
    dn=case_when(
      event=="admit"~1L,
      event=="discharge"~-1L,
```

```
      TRUE~0L
    ),
    occ=cumsum(dn)
  ) |>
  ggplot(aes(x=date,y=occ))+
  geom_step()+
  labs(title="hospital occupancy")+
  theme_bw()

fake_data |>
  arrange(time) |>
  select(date,test.result=result,isol,infected) |>
  pivot_longer(c(test.result,isol,infected)) |>
  filter(!is.na(value)) |>
  ggplot(aes(x=date,y=value,color=name))+
  geom_point()+
  geom_smooth()+
  guides(color="none")+
  labs(
    title="infection and isolation status, test results",
    y=""
  )+
  facet_grid(name~.)+
  theme_bw()

fake_data |>
  filter(event=="test") |>
  mutate(
    interval=cut(time,breaks=72,ordered=TRUE)
  ) |>
  select(interval,time,infected,isol,result) |>
  pivot_longer(c(infected,isol,result)) |>
  group_by(name,interval) |>
  summarize(
    time=mean(time),
    prev=mean(value),
    n=n()
  ) |>
  ungroup() |>
  ggplot(aes(x=time,y=prev,group=name,fill=name))+
  geom_col(position="dodge")+
  labs(title="infection, isolation, and detection through time")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90))

fake_data |>
  filter(event=="test") |>
  select(infected,result) |>
  count(infected,result) |>
  group_by(infected) |>
  mutate(prob=n/sum(n)) |>
  ungroup()
```

stobfun *Stateful objective functions*

Description

Convenience functions for constructing and working with stateful objective functions ('stobfun'-class objects).

Usage

```
stobfun(
  embed,
  params,
  est = character(0),
  log = character(0),
  logit = character(0),
  objfun,
  data,
  ...
)

filterfun(embed, params, filtfun, data, ...)

transf_fns(log = character(0), logit = character(0), est = character(0))

embedding(...)

## S3 method for class 'stobfun'
coef(object, ...)
```

Arguments

`embed`	embedding (see `embedding`).
`params`	vector of parameters
`est`	character: names of parameters to be estimated
`log`	character; names of parameters to log transform.
`logit`	character; names of parameters to logit transform.
`objfun`	underlying objective function
`data`	data
`...`	When furnished to `stobfun`, additional arguments are passed to `objfun`. When furnished to `embedding`, arguments define th embedding. When furnished to `coef`, additional arguments are ignored.
`filtfun`	the function that actually applies the filter
`object`	'stobfun'-class stateful objective function

Details

objfun will be called as objfun(theta,data), where theta is the nested list constructed according to the given specifications and data is the data.

Value

transf_fns returns a list of two functions. The first is the transformation to the estimation scale; the second is its inverse.

embedding returns the embedding function corresponding to the given specification.

coef(f) returns the parameter vector corresponding to the last call of the stateful objective function f.

Construction and usage of stateful objective functions

A stateful objective function is an ordinary function that can be used as an objective function in an optimization problem. In particular, it can be passed to optimizers such as optim, subplex, or nloptr. It is stateful in the sense that it remembers the argument with which it was last called.

To construct a stateful objective function, call the constructor function for the model of interest. The constructor function requires that you pass a vector of model parameters: this gives the default parameter values. It also requires that you pass the data and the names of the parameters that you wish to estimate. The constructor will return an object of class 'stobfun'.

Having constructed a 'stobfun' stateful objective function, you can pass this to any suitable optimizer. Once the optimizer has returned, **it is important that you call the function one last time, at the parameters the optimizer has returned** (see examples). This ensures that the stored parameters are those at the (putative) optimum. You can retrieve these parameters via a call to coef.

See Also

More on stateful objective functions: independent, transmission

---

transmission — *Transmission model*

Description

Model under which the force of infection in each unit is proportional to the prevalence of infection in that unit.

Usage

```
trans_homog_filter(params, data, tol = 1e-04, maxit = 10)

trans_homog_objfun(params, data, est = character(0), tol = 1e-04, maxit = 10)

trans_unit_spec_filter(params, data, tol = 1e-04, maxit = 10)
```

```
trans_unit_spec_objfun(
  params,
  data,
  est = character(0),
  tol = 1e-04,
  maxit = 10
)
```

Arguments

`params`	named vector of parameters
`data`	patient movement, isolation, and testing data
`tol`	positive scalar; convergence tolerance (mean difference).
`maxit`	scalar integer; maximum number of fixed-point iterations. If `tol` is not achieved in `maxit` or fewer iterations, an error is generated.
`est`	names of parameters to estimate

Details

The basic transmission model assumes that the force of infection on an unisolated patient is

$$\lambda = b(P_u + aP_i) + \iota,$$

where P_u, P_i are the prevalences among unisolated and isolated patients, respectively and ι represents the risk of acquisition unrelated to local prevalence. On an isolated patient, this force of infection is reduced by the factor `isol_factor`.

Parameters in the model include:

b transmission rate. In the homogeneous model, this is constant across units in the hospital; in the unit-specific model, there is one value of `b` for each unit.

lambda.out force of colonization outside the hospital

iota baseline force of colonization

gamma recovery rate

alpha,beta false positive and negative probabilities

p0 probability of infection on admission

a multiplicative effect of contact isolation on transmissibility

isol_factor multiplicative effect of contact isolation on susceptibility

`trans_homog_filter` runs a fixed-point Bernoulli filter for the transmission model with global b and gamma.

`trans_homog_objfun` is a stateful objective function for the transmission model with global b and gamma.

`trans_unit_spec_filter` runs a fixed-point Bernoulli filter for the transmission model with unit-specific b.

`trans_unit_spec_objfun` is a stateful objective function for the transmission model with unit-specific transmission rates, an out-of-hospital force of infection parameter, and recovery rates that can be different inside and outside of hospital.

See Also

More on stateful objective functions: independent, stobfun()

Examples

```
library(tidyverse)
library(hte)

set.seed(339613584)

fake_data |>
  ## filter out tests prior to day 250
  filter(
    event!="test" | time < 250,
    time < 1000
  ) |>
  select(-infected) |>
  arrange(patient,time) -> dat

trans_homog_objfun(
  params=c(
    a=1,b=0.1,gamma=0.01,p0=0.1,lambda.out=0.01,
    iota=0,isol_factor=0.2,alpha=0.02,beta=0.1
  ),
  est=c("a","b"),
  data=dat
) -> f
f(log(c(0.05,0.4)))
coef(f)

optim(
  par=log(c(0.5,0.4)),
  fn=f,
  control=list(reltol=1e-2)
) -> out
f(out$par)
coef(f)

trans_homog_filter(params=coef(f),data=dat) -> ff
ff |> filter(logLik!=0)

trans_unit_spec_objfun(
  params=c(
    a=1,
    b.A=0.1,b.B=0.1,b.C=0.1,b.D=0.1,b.E=0.1,
    gamma=0.01,p0=0.1,
    lambda.out=0.1,
    iota=0,
    isol_factor=0.2,
    alpha=0.02,beta=0.1
  ),
  data=dat
```

```
) -> f
f()
coef(f)
```

twostate	*twostate*

Description

Two-state Markov process with on and off rates

Usage

```
twostate(on, off, tf, t0 = 0)
```

Arguments

`on, off`	on and off rates
`t0, tf`	initial and final times

Index