Package ‘tsiR’

January 20, 2021

Type    Package
Title   An Implementation of the TSIR Model
Version 0.4.3
Date    2021-01-05
Description An implementation of the time-series Susceptible-Infected-Recovered (TSIR) model using a number of different fitting options for infectious disease time series data. The manuscript based on this package can be found here <doi:10.1371/journal.pone.0185528>. The method implemented here is described by Finkenstadt and Grenfell (2000) <doi:10.1111/1467-9876.00187>.
Depends R (>= 3.1.0)
License GPL-3
LazyData true
Imports ggplot2, kernlab, reshape2, grid
Contact adbecker at princeton dot edu
RoxygenNote 7.1.1
NeedsCompilation no
Author Alexander D. Becker [aut, cre], Sinead E. Morris [ctb], Ottar N. Bjornstad [ctb]
Maintainer Alexander D. Becker <adbecker@princeton.edu>
Repository CRAN
Date/Publication 2021-01-20 18:20:02 UTC

R topics documented:
corr ................................................................. 2
derivative .......................................................... 3
epitimes ............................................................ 3
estpars ........................................................... 4
jagsfilter ......................................................... 5
logcorr .......................................................... 6
Description

Plot the correlation of the true data against the fitted resimulated data.

Usage

\texttt{corr(sim)}

Arguments

\texttt{sim} The dataframe or list produced by the 'runtsir' function.
derivative  

**Description**

This function computes an 8 point derivative.

**Usage**

derivative(X, Y)

**Arguments**

<table>
<thead>
<tr>
<th>X</th>
<th>The variable to differentiate with respect to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>The function / vector to differentiate.</td>
</tr>
</tbody>
</table>

epitimes  

**Description**

The times at which we declare a new outbreak has started based on the threshold parameter.

**Usage**

epitimes(data, threshold, epi.length = 3)

**Arguments**

<table>
<thead>
<tr>
<th>data</th>
<th>The inputed data frame with the cases vector. This is the same data you put into runtsir.</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold</td>
<td>The required number of cases observed to declare it an outbreak.</td>
</tr>
<tr>
<td>epi.length</td>
<td>The required duration (in 52/IP weeks) to declare it an outbreak.</td>
</tr>
</tbody>
</table>
estpars

Description

This function computes the set up to run the TSIR model, i.e. reconstructs susceptibles and estimates beta and alpha. This can be plugged into simulatetsir.

Usage

```
estpars(
data, xreg = "cumcases", IP = 2, seasonality = "standard", regtype = "gaussian", sigmamax = 3, family = "gaussian", link = "identity", userYhat = numeric(), alpha = NULL, sbar = NULL, printon = F)
```

Arguments

data The data frame containing cases and interpolated births and populations.
xreg The x-axis for the regression. Options are 'cumcases' and 'cumbirths'. Defaults to 'cumcases'.
IP The infectious period in weeks. This should be the same as your timestep. Defaults to 2 weeks.
seasonality The type of contact to use. Options are standard for 52/IP point contact or schoolterm for just a two point on off contact or none for a single contact parameter. Defaults to standard.
regtype The type of regression used in susceptible reconstruction. Options are 'gaussian', 'lm' (linear model), 'spline' (smooth.spline with 2.5 degrees freedom), 'loess' (with f = 2/3, iter = 1), 'loess' (degree 1), and 'user' which is just a user inputted vector. Defaults to 'gaussian' and if that fails then defaults to loess.
sigmamax The inverse kernel width for the gaussian regression. Default is 3. Smaller, stochastic outbreaks tend to need a lower sigma.
family The family in the GLM regression. One can use any of the GLM ones, but the options are essentially 'poisson' (with link='log'), 'gaussian' (with link='log' or 'identity'), or 'quasipoisson' (with link='log'). Default is 'gaussian'.
The link function used with the glm family. Options are link='log' or 'identity'. Default is 'identity'. To include some bayesian approaches. For 'bayesglm' we use a gaussian prior with mean 1e-4.

The inputed regression vector if regtype='user'. Defaults to NULL.

The mixing parameter. Defaults to NULL, i.e. the function estimates alpha.

The mean number of susceptibles. Defaults to NULL, i.e. the function estimates sbar.

Whether to show diagnostic prints or not, defaults to FALSE.

Examples

## Not run:
require(kernlab)
London <- twentymeas[['London']] 
parms <- estpars(London)
names(parms)
sim <- simulatetsir(London,parms=parms,inits.fit=FALSE) 
plotres(sim)

## End(Not run)

Description

Used internally to filter jags results to give just the inference well use.

Usage

jagsfilter(mcmcrealts)

Arguments

mcmcrealts is the input from the jags model.
logcorr

**Description**

Plot the correlation of the true data against the fitted resimulated data.

**Usage**

\[ \text{logcorr(sim)} \]

**Arguments**

- **sim**: The dataframe or list produced by the 'runtsir' function.

maxthreshold

**Description**

A function used to optimize the threshold parameter to give the best fit to the data. Optimizes the fit based on R squared.

**Usage**

\[ \text{maxthreshold(data, nsim = 2, IP = 2, method = "deterministic", inits.fit = FALSE, parms, thresholdmin = 2, thresholdmax = 20, printon = FALSE) } \]

**Arguments**

- **data**: The time, cases, births, pop data frame.
- **nsim**: The number of simulations to do.
- **IP**: The infectious period, which should the time step of the data.
- **method**: The forward simulation method used, i.e. deterministic, negbin, pois.
- **inits.fit**: Whether or not to fit initial conditions as well. Defaults to FALSE here. This parameter is more necessary in more chaotic locations.
**mcmcestpars**

<table>
<thead>
<tr>
<th>params</th>
<th>The estimated parameters from estpars or mcmcestpars.</th>
</tr>
</thead>
<tbody>
<tr>
<td>thresholdmin</td>
<td>The minimum number of cases to be considered an outbreak.</td>
</tr>
<tr>
<td>thresholdmax</td>
<td>The max number of cases to be considered an outbreak.</td>
</tr>
<tr>
<td>printon</td>
<td>A T/F statement to print the progress.</td>
</tr>
</tbody>
</table>

**Examples**

```r
require(kernlab)
Mold <- twentymeas[["Mold"]]
plotdata(Mold)
## Not run:
parms <- estpars(data=Mold,alpha=0.97)
tau <- maxthreshold(data=Mold,parms=parms,
thresholdmin=8,thresholdmax=12,inits.fit=FALSE)
res <- simulatetsir(data=Mold,parms=parms,
epidemics='break',threshold=tau,method='negbin',inits.fit=FALSE)
plotres(res)
## End(Not run)
```

**Description**

This function computes the set up to run the TSIR model, i.e. reconstructs susceptibles and estimates beta and alpha using MCMC computations. Used the same way as estpars.

**Usage**

```r
mcmcestpars(
  data,
  xreg = "cumcases",
  IP = 2,
  regtype = "gaussian",
  sigmamax = 3,
  seasonality = "standard",
  userYhat = numeric(),
  update.iter = 10000,
  n.iter = 30000,
  n.chains = 3,
  n.adapt = 1000,
  burn.in = 100,
  sbar = NULL,
  alpha = NULL,
  alpha = NULL,
  printon = F
)
```
Arguments

- **data**: The data frame containing cases and interpolated births and populations.
- **xreg**: The x-axis for the regression. Options are 'cumcases' and 'cumbirths'. Defaults to 'cumcases'.
- **IP**: The infectious period in weeks. Defaults to 2 weeks.
- **regtype**: The type of regression used in susceptible reconstruction. Options are 'gaussian', 'lm' (linear model), 'spline' (smooth.spline with 2.5 degrees freedom), 'lowess' (with f = 2/3, iter = 1), 'loess' (degree 1), and 'user' which is just a user inputed vector. Defaults to 'gaussian' and if that fails then defaults to loess.
- **sigmamax**: The inverse kernel width for the gaussian regression. Default is 3. Smaller, stochastic outbreaks tend to need a lower sigma.
- **seasonality**: The type of contact to use. Options are standard for 52/IP point contact or schoolterm for just a two point on off contact or none for a single contact parameter. Defaults to standard.
- **userYhat**: The inputed regression vector if regtype='user'. Defaults to NULL.
- **update.iter**: Number of MCMC iterations to use in the update aspect. Default is 10000.
- **n.iter**: Number of MCMC iterations to use. Default is 30000.
- **n.chains**: Number of MCMC chains to use. Default is 3.
- **n.adapt**: Adaptive number for MCMC. Default is 1000.
- **burn.in**: Burn in number. Default is 100.
- **sbar**: The mean number of susceptibles. Defaults to NULL, i.e. the function estimates sbar.
- **alpha**: The mixing parameter. Defaults to NULL, i.e. the function estimates alpha.
- **printon**: Whether to show diagnostic prints or not, defaults to FALSE.

Description

This function runs the TSIR model using a MCMC estimation. The susceptibles are still reconstructed in the same way as the regular tsir model, however beta, alpha, and sbar (or whatever combination you enter) are estimated using rjargs.

Usage

```r
mcmctsir(
  data,
  xreg = "cumcases",
  IP = 2,
  nsim = 100,
  regtype = "gaussian",
)```
mcmctsir

```r
sigmamax = 3,
userYhat = numeric(),
update.iter = 10000,
n.iter = 30000,
n.chains = 3,
n.adapt = 1000,
burn.in = 100,
method = "deterministic",
epidemics = "cont",
pred = "forward",
seasonality = "standard",
inits.fit = FALSE,
threshold = 1,
sbar = NULL,
alpha = NULL,
add.noise.sd = 0,
mul.noise.sd = 0,
printon = F
)
```

**Arguments**

- **data**: The data frame containing cases and interpolated births and populations.
- **xreg**: The x-axis for the regression. Options are 'cumcases' and 'cumbirths'. Defaults to 'cumcases'.
- **IP**: The infectious period in weeks. Defaults to 2 weeks.
- **nsim**: The number of simulations to do. Defaults to 100.
- **regtype**: The type of regression used in susceptible reconstruction. Options are 'gaussian', 'lm' (linear model), 'spline' (smooth.spline with 2.5 degrees freedom), 'lowess' (with f = 2/3, iter = 1), 'loess' (degree 1), and 'user' which is just a user inputed vector. Defaults to 'gaussian' and if that fails then defaults to loess.
- **sigmamax**: The inverse kernal width for the gaussian regression. Default is 3. Smaller, stochastic outbreaks tend to need a lower sigma.
- **userYhat**: The inputed regression vector if regtype='user'. Defaults to NULL.
- **update.iter**: Number of MCMC iterations to use in the update aspect. Default is 10000.
- **n.iter**: Number of MCMC iterations to use. Default is 30000.
- **n.chains**: Number of MCMC chains to use. Default is 3.
- **n.adapt**: Adaptive number for MCMC. Default is 1000.
- **burn.in**: Burn in number. Default is 100.
- **method**: The type of next step prediction used. Options are 'negbin' for negative binomial, 'pois' for poisson distribution, and 'deterministic'. Defaults to 'deterministic'.
- **epidemics**: The type of data splitting. Options are 'cont' which doesn't split the data up at all, and 'break' which breaks the epidemics up if there are a lot of zeros. Defaults to 'cont'.
pred
The type of prediction used. Options are 'forward' and 'step-ahead'. Defaults to 'forward'.

seasonality
The type of contact to use. Options are standard for 52/IP point contact or schoolterm for just a two point on off contact or none for a single contact parameter. Defaults to standard.

inits.fit
Whether or not to fit initial conditions using simple least squares as well. Defaults to FALSE. This parameter is more necessary in more chaotic locations.

threshold
The cut off for a new epidemic if epidemics = 'break'. Defaults to 1.

sbar
The mean number of susceptibles. Defaults to NULL, i.e. the function estimates sbar.

alpha
The mixing parameter. Defaults to NULL, i.e. the function estimates alpha.

add.noise.sd
The sd for additive noise, defaults to zero.

mul.noise.sd
The sd for multiplicative noise, defaults to zero.

printon
Whether to show diagnostic prints or not, defaults to FALSE.

---

plotbeta

**Description**

Plots the inferred beta with confidence intervals (when they can be calculated)

**Usage**

plotbeta(dat)

**Arguments**

dat
the list produced from the runtsir, mcmctsir, and simulatetsir function.

---

plotbreaks

**Description**

Plots the cases data with a line whenever the forward simulation is seeded using the real data.

**Usage**

plotbreaks(data, threshold)

**Arguments**

data
Data frame with the cases vector.

threshold
The epidemic threshold, i.e. the number of cases required to spark a new outbreak in the model.
### plotcases

**Description**
Plots just the cases data.

**Usage**
```r
plotcases(data)
```

**Arguments**
- `data`: The data frame with cases.

### plotcomp

**Description**
Plots just the comparison of the forward simulation fit to the data.

**Usage**
```r
plotcomp(sim, errtype = "95", max.plot = 10)
```

**Arguments**
- `sim`: is list produced by runtsir or mcmctsir
- `errtype`: is the type of error bands to show. Defaults to '95' for 95 percent CI, the other option is 'sd' to standard deviation.
- `max.plot`: the number of individual stochastic simulations to plot. Defaults to 10.

### plotdata

**Description**
Plots the cases data as well as birth and population dynamics.

**Usage**
```r
plotdata(data)
```

**Arguments**
- `data`: The dataframe with time, cases, births, and pop.
**plotforward**

**Description**
Plots the forward simulation from the TSIR model.

**Usage**
```r
plotforward(dat, inverse = F)
```

**Arguments**
- `dat`: the list produced from the runtsir, mcmctsir, and simulatetsir function.
- `inverse`: a TRUE or FALSE option to plot the forward simulate negative (TRUE) or positive (FALSE). Defaults to FALSE.

---

**plotLLE**

**Description**
Function to plot the Local Lyapunov Exponents. The output is of class ggplot2 so you can add standard ggplot2 options to it if desired.

**Usage**
```r
plotLLE(LLE)
```

**Arguments**
- `LLE`: The output from TSIR_LLE

**Examples**

```r
## Not run:
require(kernlab)
require(ggplot2)
require(kernlab)
London <- twentymeas$London
## just analyze the biennial portion of the data
London <- subset(London, time > 1950)
## define the interval to be 2 weeks
```
IP <- 2

## first estimate parameters from the London data
parms <- estpars(data=London, IP=2, regtype='gaussian', family='poisson', link='log')

## look at beta and alpha estimate
plotbeta(parms)

## simulate the fitted parameters
sim <- simulatetsir(data=London, parms=parms, IP=2, method='deterministic', nsim=2)

## now let's predict forward 200 years using the mean birth rate,
## starting from rough initial conditions
times <- seq(1965, 2165, by = 1/ (52/IP))
births <- rep(mean(London$births), length(times))
S0 <- parms$sbar
I0 <- 1e-5*mean(London$pop)

pred <- predicttsir(times=times, births=births,
beta=parms$contact$beta, alpha=parms$alpha,
S0=S0, I0=I0,
nsim=50, stochastic=T)

## take the last 10 years
pred <- lapply(pred, function(x) tail(x, 52/IP * 20) )

## now compute the Lyapunov Exponent for the simulate and predicted model

simLE <- TSIR_LE(
  time=sim$res$time,
  S=sim$simS$mean,
  I=sim$res$mean,
  alpha=sim$alpha,
  beta=sim$contact$beta,
  IP=IP
)

predLE <- TSIR_LE(
  time=pred$I$time,
  S=pred$S$X3,
  I=pred$I$X3,
  alpha=parms$alpha,
  beta=parms$contact$beta,
  IP=IP
)

simLE$LE
predLE$LE

simLLE <- TSIR_LLE(simLE)
predLLE <- TSIR_LLE(predLE)

plotLLE(simLLE)
plotLLE(predLLE)
## End(Not run)

plotregression

Description

Plots the cumulative cases - cumulative births data and regression fit

Usage

plotregression(dat)

Arguments

dat the list produced from the runtsir, mcmctsir, and simulatetsir function.

plotres

Description

Plots diagnostics and results of the runtsir function.

Usage

plotres(dat, max.plot = 10)

Arguments

dat the list produced from the runtsir, mcmctsir, and simulatetsir function.
max.plot the number of individual stochastic simulations to plot. Defaults to 10.
plotrho

Description
Plots the inferred reporting rate, rho

Usage
plotrho(dat)

Arguments
dat the list produced from the runtsir, mcmctsir, and simulatetsir function.

plotsbar

Description
Plots the profile log likelihood calculation for inferred sbar

Usage
plotsbar(dat)

Arguments
dat the list produced from the runtsir, mcmctsir, and simulatetsir function.

predicttsir

Description
Function to predict incidence and susceptibles using the tsir model. This is different than simulatetsir as you are inputting parameters as vectors. The output is a data frame I and S with mean and confidence intervals of predictions.

Usage
predicttsir(times, births, beta, alpha, S0, I0, nsim, stochastic)
predicttsir

Arguments

- **times**: The time vector to predict the model from. This assumes that the time step is equal to IP.
- **births**: The birth vector (of length length(times) or a single element) where each element is the births in that given (52/IP) time step.
- **beta**: The length(52/IP) beta vector of contact.
- **alpha**: A single numeric which acts as the homogeniety parameter.
- **S₀**: The starting initial condition for S. This should be greater than one, i.e. not a fraction.
- **I₀**: The starting initial condition for I. This should be greater than one, i.e. not a fraction.
- **nsim**: The number of simulations to perform.
- **stochastic**: A TRUE / FALSE argument where FALSE is the deterministic model, and TRUE is a negative binomial distribution.

Examples

```r
## Not run:
require(kernlab)
require(ggplot2)
require(kernlab)
require(tsiR)
London <- twentymeas$London

London <- subset(London, time > 1950)

IP <- 2
## first estimate paramters from the London data
parms <- estpars(data=London, IP=2, regtype='gaussian')

plotbeta(parms)

## now lets predict forward 20 years using the mean birth rate,
## starting from rough initial conditions
births <- min(London$births)
times <- seq(1965,1985, by = 1/ (52/IP))
S₀ <- parms$sbar
I₀ <- 1e-5*mean(London$pop)

pred <- predicttsir(times=times, births=births,
                      beta=parms$contact$beta, alpha=parms$alpha,
                      S₀=S₀, I₀=I₀,
                      nsim=50, stochastic=T)

## plot this prediction
ggplot(pred$I,aes(time,mean))+geom_line()+geom_ribbon(aes(ymin=low,ymax=high),alpha=0.3)
```
### residual.births

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the residuals for when X is the cumulative births. Used internally.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>residual.births(rho, Yhat, Y)</td>
</tr>
<tr>
<td>Arguments</td>
<td><strong>rho</strong>  The reporting rate, used to get units correct.</td>
</tr>
<tr>
<td></td>
<td><strong>Yhat</strong> The fitted regression line.</td>
</tr>
<tr>
<td></td>
<td><strong>Y</strong>    The cumulative cases.</td>
</tr>
</tbody>
</table>

### residual.cases

<table>
<thead>
<tr>
<th>Description</th>
<th>Computes the residuals for when X is the cumulative cases. Used internally.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>residual.cases(Yhat, Y)</td>
</tr>
<tr>
<td>Arguments</td>
<td><strong>Yhat</strong> The fitted regression line.</td>
</tr>
<tr>
<td></td>
<td><strong>Y</strong>    The cumulative births.</td>
</tr>
</tbody>
</table>
Description

This function runs the TSIR model.

Usage

runtsir(
  data,
  xreg = "cumcases",
  IP = 2,
  nsim = 10,
  regtype = "gaussian",
  sigmamax = 3,
  userYhat = numeric(),
  alpha = NULL,
  sbar = NULL,
  family = "gaussian",
  link = "identity",
  method = "deterministic",
  inits.fit = FALSE,
  epidemics = "cont",
  pred = "forward",
  threshold = 1,
  seasonality = "standard",
  add.noise.sd = 0,
  mul.noise.sd = 0,
  printon = F,
  fit = NULL,
  fittype = NULL
)

Arguments

data The data frame containing cases and interpolated births and populations.

xreg The x-axis for the regression. Options are 'cumcases' and 'cumbirths'. Defaults to 'cumcases'.

IP The infectious period in weeks. Defaults to 2 weeks.

nsim The number of simulations to do. Defaults to 100.

regtype The type of regression used in susceptible reconstruction. Options are 'gaussian', 'lm' (linear model), 'spline' (smooth.spline with 2.5 degrees freedom), 'loess' (with \( f = 2/3 \), \( \text{iter} = 1 \)), 'loess' (degree 1), and 'user' which is just a user inputed vector. Defaults to 'gaussian' and if that fails then defaults to loess.
### Examples

```r
require(kernlab)
London <- twentymeas[["London"]]
## Not run:
plotdata(London)
res <- runtsir(data=London, method="pois", nsim=10, IP=2, inits.fit=FALSE)
plotres(res)
## End(Not run)
```
Description

This function just simulates the forward prediction given the data and a parms list generated from estpars or mcmcestpars.

Usage

```r
simulatetsir(
  data,
  nsim = 100,
  IP = 2,
  parms,
  method = "deterministic",
  epidemics = "cont",
  pred = "forward",
  threshold = 1,
  inits.fit = FALSE,
  add.noise.sd = 0,
  mul.noise.sd = 0
)
```

Arguments

data: The data frame containing cases and interpolated births and populations.
nsim: The number of simulations to do. Defaults to 100.
IP: The infectious period. Defaults to 2.
parms: Either the parameters estimated by estpars or mcmcestpars, or a list containing beta, rho, Z, sbar, alpha, X, Y, Yhat, contact, alphalow, alphahigh, loglik, pop vectors.
method: The type of next step prediction used. Options are 'negbin' for negative binomial, 'pois' for poisson distribution, and 'deterministic'. Defaults to 'deterministic'.
epidemics: The type of data splitting. Options are 'cont' which doesn't split the data up at all, and 'break' which breaks the epidemics up if there are a lot of zeros. Defaults to 'cont'.
pred: The type of prediction used. Options are 'forward' and 'step-ahead'. Defaults to 'forward'.
threshold: The cut off for a new epidemic if epidemics = 'break'. Defaults to 1.
inits.fit: Whether or not to fit initial conditions using simple least squares as well. Defaults to FALSE. This parameter is more necessary in more chaotic locations.
add.noise.sd: The sd for additive noise, defaults to zero.
mul.noise.sd: The sd for multiplicative noise, defaults to zero.
**tsiRdata**

**Description**

A function to take in time cases births and pop vectors (of any lengths) and interpolate them using the given infectious period.

**Usage**

```r
tsiRdata(time, cases, births, pop, IP = 2)
```

**Arguments**

- **time**: The time vector.
- **cases**: The cases vector.
- **births**: The births vector.
- **pop**: The population vector.
- **IP**: The infectious period (in weeks) to discretize to. Defaults to 2.

---

**TSIR_LE**

**Description**

A function to calculate the Lyapunov Exponent (LE) from the TSIR model.

**Usage**

```r
TSIR_LE(time, S, I, alpha, beta, IP)
```

**Arguments**

- **time**: The time vector from the data or simulated data.
- **S**: The S output from the simulated or predicted TSIR model.
- **I**: The I output from the simulated or predicted TSIR model.
- **alpha**: The homogeneity parameter from the simulated or predicted TSIR model.
- **beta**: The inferred contact rate, use beta = contact$beta where contact is an output from runtsir or simulatsir.
- **IP**: The generation interval of the pathogen (in weeks).
Examples

```r
## Not run:
require(kernlab)
require(ggplot2)
require(kernlab)
London <- twentymeas$London
## just analyze the biennial portion of the data
London <- subset(London, time > 1950)

## define the interval to be 2 weeks
IP <- 2

## first estimate parameters from the London data
parms <- estpars(data=London, IP=2, regtype='gaussian', family='poisson', link='log')

## look at beta and alpha estimate
plotbeta(parms)

## simulate the fitted parameters
sim <- simulatetsir(data=London, parms=parms, IP=2, method='deterministic', nsim=2)

## now lets predict forward 200 years using the mean birth rate,
## starting from rough initial conditions
	times <- seq(1965, 2165, by = 1/(52/IP))
births <- rep(mean(London$births), length(times))
	S0 <- parms$sbar
	I0 <- 1e-5*mean(London$pop)
	pred <- predicttsir(times=times, births=births,
	beta=parms$contact$beta, alpha=parms$alpha,
	S0=S0, I0=I0,
	nsim=50, stochastic=T)

## take the last 10 years
pred <- lapply(pred, function(x) tail(x, 52/IP * 20 )

## now compute the Lyapunov Exponent for the simulate and predicted model
simLE <- TSIR_LE(

time=sim$res$time,
S=sim$simS$mean,
I=sim$res$mean,
alpha=sim$alpha,
beta=sim$contact$beta,
IP=IP
)

predLE <- TSIR_LE(

time=pred$I$time,
S=pred$S$X3,
I=pred$I$X3,
```
### Description

A function to calculate the Local Lyapunov Exponent (LLE) from the TSIR model.

### Usage

```r
TSIR_LLE(LE, m = 1)
```

### Arguments

- `LE` : The output of `TSIR_LE` to pass the Jacobian elements
- `m` : The window to sweep the time-varying Jacobian elements. Defaults to one.

### Examples

```r
## Not run:
require(kernlab)
require(ggplot2)
require(kernlab)
London <- twentymeas$London
## just analyze the biennial portion of the data
London <- subset(London, time > 1950)

## define the interval to be 2 weeks
IP <- 2

## first estimate parameters from the London data
parms <- estpars(data=London, IP=2, regtype='gaussian', family='poisson', link='log')

## look at beta and alpha estimate
plotbeta(parms)

## simulate the fitted parameters
```
twentymeas <- simulateSIR(data=London, parms=parms, IP=2, method='deterministic', nsim=2)

## now lets predict forward 200 years using the mean birth rate,
## starting from rough initial conditions

times <- seq(1965, 2165, by = 1/(52/IP))
births <- rep(mean(London$births), length(times))
S0 <- parms$sbar
I0 <- 1e-5*mean(London$pop)

pred <- predictSIR(times=times, births=births,
                    beta=parms$contact$beta, alpha=parms$alpha,
                    S0=S0, I0=I0,
                    nsim=50, stochastic=T)

## take the last 10 years
pred <- lapply(pred, function(x) tail(x, 52/IP * 20) )

## now compute the Lyapunov Exponent for the simulate and predicted model

simLE <- TSIR_LE(time=sim$res$time,
                  S=sim$simS$mean,
                  I=sim$res$mean,
                  alpha=sim$alpha,
                  beta=sim$contact$beta,
                  IP=IP)

predLE <- TSIR_LE(time=pred$I$time,
                  S=pred$S$X3,
                  I=pred$I$X3,
                  alpha=parms$alpha,
                  beta=parms$contact$beta,
                  IP=IP)

simLLE <- TSIR_LLE(simLE)
predLLE <- TSIR_LLE(predLE)

plotLLE(simLLE)
pplotLLE(predLLE)

## End(Not run)
Description
twentymeas is a list containing 20 dataframes with cases, births, populations. Each dataframe is a 22 year time series at biweekly (i.e. IP=2) intervals.

Usage
data("twentymeas")

Source
From Bryan Grenfell

Examples
\begin{verbatim}
  names(twentymeas)
  london <- twentymeas["London"]
  plotdata(london)
\end{verbatim}

---

Description
the function just breaks up the plot area into a grid. Called internally.

Usage
vplayout(x, y)

Arguments
\begin{verbatim}
x is the x location of the plot
y is the y location of the plot
\end{verbatim}
Index

* datasets
  * twentymeas, 24

corr, 2

derivative, 3

epistimes, 3

estpars, 4

jagsfilter, 5

logcorr, 6

maxthreshold, 6

mcmcestpars, 7

mcmctsir, 8

plotbeta, 10

plotbreaks, 10

plotcases, 11

plotcomp, 11

plotdata, 11

plotforward, 12

plotLLE, 12

plotregression, 14

plotres, 14

plotrho, 15

plotsbar, 15

predicttsir, 15

residual.births, 17

residual.cases, 17

runtsir, 18

simulatetsir, 20

TSIR.LE, 21

TSIR.LLE, 23

tsiRdata, 21

twentymeas, 24

vplayout, 25