Package ‘NHPoisson’

Type Package
Title Modelling and Validation of Non Homogeneous Poisson Processes
Version 3.1
Date 2015-03-11
Author Ana C. Cebrian <acebrian@unizar.es>
Maintainer Ana C. Cebrian <acebrian@unizar.es>
Imports car, parallel
Depends methods, stats4
Description Tools for modelling, ML estimation, validation analysis and simulation of non homogenous Poisson processes in time.
License GPL (>= 2)
NeedsCompilation no
Repository CRAN
Date/Publication 2015-03-15 10:10:47

R topics documented:

NHPoisson-package .................................................. 2
addAIC.fun ............................................................... 3
BarTxTn ................................................................. 4
CalcRes.fun .............................................................. 5
CalcResD.fun ............................................................ 7
CIdelta.fun .............................................................. 9
CItran.fun ............................................................... 10
confintAsin.fun ....................................................... 11
dropAIC.fun ........................................................... 13
emplambda.fun ........................................................ 14
emplambdaD.fun ....................................................... 15
extractAIC-methods .................................................. 17
fitPP.fun ............................................................... 17
GenEnv.fun ............................................................. 20
globalval.fun ......................................................... 21
NHPoisson-package

Statistical modelling of non homogeneous Poisson processes

Description

NHPoisson provides tools for the modelling and maximum likelihood estimation of non homogeneous Poisson processes (NHPP) in time, where the intensity is formulated as a function of (time-dependent) covariates. A comprehensive toolkit for model selection, residual analysis and diagnostic of the fitted model is also provided. Finally, it permits random generation of NHPP.

Details

Package: NHPoisson
Type: Package
Version: 3.0
Date: 2014-05-21
License: GPL (>=2)

Author(s)

Ana C. Cebrian <acebrian@unizar.es>

See Also

evir, extRemes, POT, ppstat, spatstat, yuima
addAIC.fun

Calculate the AIC for all one-covariate additions to the current model

Description

This function fits all models that differ from the current model by adding a single covariate from those supplied, and calculates their AIC value. It selects the best covariate to be added to the model, according to the AIC.

Usage

addAIC.fun(mlePP, covariatesAdd, startAdd = NULL, modSim = FALSE,...)

Arguments

mlePP A "mlePP"-class object; usually the output from fitPP.fun. It defines the current model. The fitted model cannot include fixed parameters.
covariatesAdd Matrix of the potential covariates to be added to the model; each column must contain a covariate.
startAdd Optional. The vector of initial values for the estimation algorithm of the coefficients of each potential covariate. If it is NULL, initial values equal to 0 are used. Remark that in contrast to argument start of fitPP.fun, startAdd is a numeric vector not a list.
modSim Logical flag. If it is FALSE, information about the process is shown on the screen. For automatic selection processes, the option TRUE should be preferred.
...
Further arguments to pass to AIC, for example the constant k for the AIC calculation.

Details

The definition of AIC uses constant k=2, but a different value k can be passed as an additional argument. The best covariate to be added is the one which leads to the model with the lowest AIC value and it improves the current model if the new AIC is lower than the current one.

Value

A list with the following components

AICadd Vector of the AIC values obtained from adding to the current model each covariate in covariatesAdd.
posminAIC An integer indicating the number of the column of covariatesAdd with the covariate leading to the minimum AIC.
namecov Name of the covariate leading to the minimum AIC.
AICcurrent AIC value of the current (initial) model.
newCoef A (named) list with the initial value for the coefficient of the best covariate to be added. It is used in stepAICmle.fun.
See Also

dropAIC.fun, stepAICmle.fun, LRTpv.fun

Examples

data(BarTxTn)

BarEv<-POTevents.fun(T=BarTxTn$Tx, thres=318,
date=cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$dia))

#The initial model contains only the intercept
mod1Bind<-fitPP.fun(covariates=NULL, posE=BarEv$Px, inndat=BarEv$inndat,
tit='BAR Intercept ', start=list(b0=1))
#the potential covariates
covB<-cbind(cos(2*pi*BarTxTn$dia/365), sin(2*pi*BarTxTn$dia/365),
BarTxTn$TTx,BarTxTn$Txm31,BarTxTn$Txm31**2)
dimnames(covB)<-list(NULL,c('cos','sin','TTx','Txm31', 'Txm31**2'))
aux<-addAIC.fun(mod1Bind, covariatesAdd=covB)

BarTxTn

Barcelona temperature data

Description

Barcelona daily temperature series during the summer months (May, June, July, August and September) from 1951 to 2004.

Usage

data(BarTxTn)

Details

Variables
dia: Postion of the day in the year, from 121 (1st of May) to 253 (30th of September).
mes: Month of the year, from 5 to 9.
ano: Year, from 1951 to 2004.
dia mes: Position of the day in the month, from 1 to 30 or 31.
Tx: Daily maximum temperature.
Tn: Daily minimum temperature.
T xm31: Local maximum temperature signal. Lowess of Tx with a centered window of 31 days.
T xm15: Local maximum temperature signal. Lowess of Tx with a centered window of 15 days.
T nm31: Local minimum temperature signal. Lowess of Tn with a centered window of 31 days.
Tn15: Local minimum temperature signal. Lowess of Tn with a centered window of 15 days.
TTx: Long term maximum temperature signal. Lowess of Tx with a centered 40% window.
TTn: Long term minimum temperature signal. Lowess of Tn with a centered 40% window.

References

Examples
```
data(BarTxTn)
```

CalcRes.fun	*Calculate NHPP residuals on overlapping intervals*

Description
This function calculates raw and scaled residuals of a NHPP based on overlapping intervals. The scaled residuals can be Pearson or any other type of scaled residuals defined by the function \( h(t) \).

Usage
```
calcresNfun(mlePP, lint, h = NULL, typeRes = NULL)
```

Arguments
- `mlePP`: An object of class `mlePP-class`; usually, the output from `fitPP.fun`.
- `lint`: Length of the intervals to calculate the residuals.
- `h`: Optional. Weight function to calculate the scaled residuals. By default, Pearson residuals with \( h(t) = 1/\sqrt{\hat{\lambda}(t)} \) are calculated.
- `typeRes`: Optional. Label indicating the type of scaled residuals. By default, Pearson residuals are calculated and label is ‘Pearson’.

Details
The raw residuals are based on the increments of the raw process \( R(t) = N_t - \int_0^t \hat{\lambda}(u)du \) in overlapping intervals \((l_1, l_2)\) centered on \( t \):

\[
r'(l_1, l_2) = R(l_2) - R(l_1) = \sum_{t_i \in (l_1, l_2)} I_{t_i} - \int_{l_1}^{l_2} \hat{\lambda}(u)du.
\]

Residuals \( r'(l_1, l_2) \) are made ‘instantaneous’ dividing by the length of the intervals (specified by the argument `lint`), \( r(l_1, l_2) = r'(l_1, l_2)/(l_2 - l_1) \). A residual is calculated for each time in the observation period.
The function also calculates the residuals scaled with the function $h(t)$

$$r_{sca}(l_1, l_2) = \sum_{t_i \in (l_1, l_2)} h(t_i) - \int_{l_1}^{l_2} h(u)\hat{\lambda}(u)du.$$

By default, Pearson residuals with $h(t) = 1/\sqrt{\hat{\lambda}(t)}$ are calculated.

**Value**

A list with elements

- `RawRes` Numeric vector of the raw residuals.
- `ScaRes` A list with elements `ScaRes` (vector of the scaled residuals) and `typeRes` (name of the type of scaled residuals).
- `emplambda` Numeric vector of the empirical estimator of the PP intensity on the considered intervals.
- `fittedlambda` Numeric vector of the sum of the intensities $\hat{\lambda}(t)$ on the considered intervals, divided by the length of the interval.
- `lintV` Numeric vector of the exact length of each interval. The exact length is defined as the number of observations in each interval used in the estimation (observations with `inidat`=1).
- `lint` Input argument.
- `typeI` Label indicating the type of intervals used to calculate the residuals, 'Overlapping'.
- `h` Input argument.
- `mlePP` Input argument.

**References**


**See Also**

`unifresNfun`, `graphresNfun`
Examples

```r
X1<-rnorm(1000)
X2<-rnorm(1000)

modE<-fitPP.fun(tind=TRUE,covariates=cbind(X1,X2),
    posE=round(runif(40,1,1000)), inddat=rep(1,1000),
    tim=c(1:1000), tit="Simulated example",start=list(b0=1,b1=0,b2=0),
    dplot=FALSE,modCI=FALSE,modSim=TRUE)

#Residuals, based on overlapping intervals of length 50, from the fitted NHPP modE
ResE<-CalcRes.fun(mlePP=modE, lint=50)
```

CalcResD.fun

Calculate NHPP residuals on disjoint intervals

Description

This function calculates raw and scaled residuals of a NHPP based on disjoint intervals. The scaled residuals can be Pearson or any other type of scaled residuals defined by the function \( h(t) \).

Usage

```r
CalcResD.fun(mlePP, h = NULL, nint = NULL, lint = NULL, typeRes = NULL,
    modSim = "FALSE")
```

Arguments

- `mlePP` An object of class `mlePP-class`; usually, the output from `fitPP.fun`.
- `lint` Optional. Length of the intervals to calculate the residuals.
- `h` Optional. Weight function to calculate the scaled residuals. By default, Pearson residuals with \( h(t) = 1/\sqrt{\hat{\lambda}(t)} \) are calculated.
- `typeRes` Optional. Label indicating the type of scaled residuals. By default, Pearson residuals are calculated and label is 'Pearson'.
- `modSim` Logical flag. If it is FALSE, some information on the intervals is shown on the screen.
- `nint` Number of intervals used to calculate the residuals. Intervals with the same length are considered. Only one of lint or nint must be specified.
Details

The intervals used to calculate the residuals can be specified either by nint or lint; only one of the arguments must be provided. If nint is specified, intervals of equal length are calculated. The raw residuals are based on the increments of the raw process $R(t) = N_t - \int_0^t \hat{\lambda}(u) du$ in disjoint intervals $(l_1, l_2)$ centered on $t$:

$$r'(l_1, l_2) = R(l_2) - R(l_1) = \sum_{t_i \in (l_1, l_2)} I_{t_i} - \int_{l_1}^{l_2} \hat{\lambda}(u) du.$$

Residuals $r'(l_1, l_2)$ are made 'instantaneous' dividing by the length of the intervals (specified by the argument lint), $r(l_1, l_2) = r'(l_1, l_2)/(l_2 - l_1)$.

The function also calculates the residuals scaled with the function $h(t)$

$$r_{sca}(l_1, l_2) = \sum_{t_i \in (l_1, l_2)} h_{t_i} - \int_{l_1}^{l_2} h(u) \hat{\lambda}(u) du.$$

By default, Pearson residuals with $h(t) = 1/\sqrt{\hat{\lambda}(t)}$ are calculated.

Value

A list with elements

- **RawRes**: Numeric vector of the raw residuals.
- **ScaRes**: A list with elements ScaRes (vector of the scaled residuals) and typeRes (name of the type of scaled residuals).
- **emplambda**: Numeric vector of the empirical estimator of the PP intensity on the considered intervals.
- **fittedlambda**: Numeric vector of the sum of the intensities $\hat{\lambda}(t)$ on the considered intervals, divided by the length of the interval.
- **lintV**: Numeric vector of the exact length of each interval. The exact length is defined as the number of observations in each interval used in the estimation (observations with inddat=1).
- **lint**: Input argument.
- **nint**: Input argument.
- **pm**: Numeric vector of the mean point of the intervals.
- **typei**: Label indicating the type of intervals used to calculate the residuals, 'Disjoint'.
- **h**: Input argument.
- **mlePP**: Input argument.
References


See Also

`CalcRes.fun, unifres.fun, graphres.fun`

Examples

```r
X1<-rnorm(1000)
X2<-rnorm(1000)

modE<-fitPP.fun(tind=TRUE,covariates=cbind(X1,X2),
posE=round(runif(40,1,1000)), inddat=rep(1,1000),
tim=c(1:1000), tit="Simulated example",start=list(b0=1,b1=0,b2=0),
doPlots=FALSE,modCl=FALSE,modSim=TRUE)

#Residuals, based on 20 disjoint intervals of length 50, from the fitted NHPP modE

ResDE<-CalcResD.fun(mlePP=modE,lint=50)
```

CI<--length(resDE) xrange<-range(resDE)
CIdelta<--matrix(0,CI,2)
for(i in 1:CI){
 CIdelta[i,1]<--resDE[i]
 CIdelta[i,2]<--resDE[i] + confint[i]
}

**CI<--length(resDE) xrange<--range(resDE)**

**CIdelta<--matrix(0,CI,2)**

**for(i in 1:CI){**

**CIdelta[i,1]<--resDE[i]**

**CIdelta[i,2]<--resDE[i] + confint[i]**

**}**

**Description**

Given the $\hat{\beta}$ covariance matrix (or its estimation), an approximate confidence interval for each $\lambda(t)$ is calculated using the delta method.

**Usage**

```r
CI<--length(resDE) xrange<--range(resDE)
CIdelta<--matrix(0,CI,2)
for(i in 1:CI){
 CIdelta[i,1]<--resDE[i]
 CIdelta[i,2]<--resDE[i] + confint[i]
}
```

CI<--length(resDE) xrange<--range(resDE)
CIdelta<--matrix(0,CI,2)
for(i in 1:CI){
 CIdelta[i,1]<--resDE[i]
 CIdelta[i,2]<--resDE[i] + confint[i]
}

**Usage**

```r
CI<--length(resDE) xrange<--range(resDE)
CIdelta<--matrix(0,CI,2)
for(i in 1:CI){
 CIdelta[i,1]<--resDE[i]
 CIdelta[i,2]<--resDE[i] + confint[i]
}
```
Arguments

VARbeta (Estimated) Covariance matrix of the \( \hat{\beta} \) parameter vector.
lambdafit Numeric vector of fitted values of the PP intensity \( \hat{\lambda}(t) \).
covariates Matrix of covariates to estimate the PP intensity.
clevel Confidence level of the confidence intervals. A value in the interval (0,1).

Value

A list with elements

LIlambda Numeric vector of the lower values of the intervals.
UIlambda Numeric vector of the upper values of the intervals.
lambdafit Input argument.

Note

fitPP.fun calls CIcdfiNdelta.fun when the argument is CIty='Delta'.

References


See Also

CItran.fun, fitPP.fun, VARbeta.fun

Examples

aux<-CIcdfiNdelta.fun(VARbeta=0.01, lambdafit=exp(rnorm(100)), covariates=matrix(rep(1,100)), clevel=0.95)

CItran.fun

Confidence intervals for \( \lambda(t) \) based on transformation

Description

Given the \( \hat{\beta} \) covariance matrix (or its estimation), an approximate confidence interval for each \( \lambda(t) = \exp(\nu(t)) \) is calculated using a transformation of the confidence interval for the linear predictor \( \nu(t) = X(t)\beta \). The transformation is \( \exp(I_i) \), where \( I_i \) are the confidence limits of \( \nu(t) \).

Usage

CItran.fun(VARbeta, lambdafit, covariates, clevel = 0.95)
Arguments

VARbeta (Estimated) Covariance matrix of the $\hat{\beta}$ parameter vector.
lambdafit Numeric vector of fitted values of the PP intensity $\hat{\lambda}(t)$.
covariates Matrix of covariates to estimate the PP intensity.
clevel Confidence level of the confidence intervals. A value in the interval (0,1).

Value

A list with elements

LILambda Numeric vector of the lower values of the intervals.
UILambda Numeric vector of the upper values of the intervals.
lambdafit Input argument.

Note

fitPP.fun calls CITran.fun when the argument is CIty='Transf'.

References


See Also

CIdelta.fun, fitPP.fun, VARbeta.fun

Examples

aux<-CITran.fun(VARbeta=0.01, lambdafit=exp(rnorm(100)), covariates=matrix(rep(1,100)), clevel=0.95)

confintAsin.fun Compute confidence intervals for the $\beta$ parameters

Description

This function computes confidence intervals for the $\beta$ parameters.

Usage

confintAsin.fun(mlePP, level = 0.95)
confintAsin.fun

Arguments

mlePP A "mlePP"-class object; usually the output from fitPP.fun.
level The confidence level required for the intervals.

Details

The confidence intervals calculated by this function are based on the asymptotic normal approximation of the MLE of the $\beta$ parameters, that is $(\hat{\beta} - z_{(1-\alpha/2)} s.e.(\hat{\beta}), \hat{\beta} + z_{(1-\alpha/2)} s.e.(\hat{\beta}))$ with $\alpha = 1 - \text{level}$

Value

A matrix with two columns, the first contains the lower limits of the confidence intervals of all the parameters and the second the upper limits.

References


See Also

confint, VARbeta.fun

Examples

data(Bartxtn)
covB<cbind(cos(2*pi*Bartxtn$dia/365), sin(2*pi*Bartxtn$dia/365), Bartxtn$TTx,Bartxtn$Txm31,Bartxtn$Txm31**2)
BarEv<-POTevents.fun(T=Bartxtn$Tx, thres=318, date=cbind(Bartxtn$ano,Bartxtn$mes,Bartxtn$dia))
modB<fitPP.fun(covariates=covB, posE=BarEv$Px, inddat=BarEv$inddat, tit="BAR Tx; cos, sin, TTx, Txm31, Txm31**2", start=list(b0=-100,b1=1,b2=-1,b3=0,b4=0,b5=0))
confintAsin.fun(modB)
Calculate the AIC for all one-covariate deletions from the current model

Description

This function fits all models obtained from the current model by deleting one covariate (except the intercept), and calculates their AIC value. It selects the best covariate to be deleted, according to the AIC value.

Usage

dropAIC.fun(mlePP, modSim = FALSE,...)

Arguments

- mlePP: A "mlePP"-class object; usually the output from fitPP.fun. It defines the current model. The fitted model cannot include fixed parameters.
- modSim: Logical flag. If it is FALSE, information about the process is shown on the screen. For automatic selection processes, the option TRUE should be preferred.
- ...: Further arguments to pass to AIC, for example the constant k for the AIC calculation.

Details

The definition of AIC uses constant k=2, but a different value k can be passed as an additional argument. The best covariate to be deleted is the one whose deletion leads to the model with the lowest AIC value and it improves the current model if the new AIC is lower than the current one.

Value

A list with the following components

- AICadd: Vector of the AIC values obtained from deleting each covariate of the current model.
- posminAIC: An integer indicating the number of the column of the covariates matrix with the covariate leading to the minimum AIC.
- namecov: Name of the covariate leading to the minimum AIC.
- AICcurrent: AIC value of the current (initial) model.

References

See Also

addAIC.fun, stepAICmle.fun, LRTpv.fun

Examples

data(BarTxTn)

BarEv<-POTevents.fun(T=BarTxTn$Tx, thres=318,
date=cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$dia))

covB<-cbind(cos(2*pi*BarTxTn$dia/365), sin(2*pi*BarTxTn$dia/365),
BarTxTn$TTx,BarTxTn$Txm31,BarTxTn$Txm31**2)

dimnames(covB)<-list(NULL,c('cos','sin','TTx','Txm31', 'Txm31**2'))

mod1B<-fitPP.fun(covariates=covB, posE=BarEv$Px, inddat=BarEv$inddat,
tit="BAR Tx; cos, sin, TTx, Txm31, Txm31**2",
start=list(b0=-100,b1=1,b2=10,b3=0,b4=0,b5=0))

aux<-dropAIC.fun(mod1B)

---

describe lambda.fun

**Empirical occurrence rates of a NHPP on overlapping intervals**

Description

This function calculates the empirical occurrence rates of a point process on overlapping intervals. The empirical rate centered in each time of the observation period is calculated using intervals of a given length. A plot of the empirical rate over time can be performed optionally.

Usage

emplambda.fun(posE, t, lint, plotEmp = TRUE, inddat = NULL, tit ="",
scax = NULL, scay = NULL)

Arguments

- **posE**: Numeric vector of the position of the occurrence points of the NHPP (or any point process in time).
- **t**: Time index of the observation period. The simplest option is 1,...,n with n the length of the period.
- **lint**: Length of the intervals used to calculate the rates.
- **plotEmp**: Logical flag. If it is TRUE, a plot of the empirical rate is carried out.
- **inddat**: Optional. Index vector equal to 1 for the observations used in the estimation process. By default, all the observations are considered, see POTevents.fun.
emplambdaD.fun

```r
emplambdaD.fun

  tit    Character string. A title for the plot.
  scax   Optional. A two element vector indicating the x-scale for the plot.
  scay   Optional. A two element vector indicating the y-scale for the plot.

Value

A list with elements

  emplambda Vector of the empirical rates.
  lint     Input argument.
```

See Also

emplambdaD.fun, fitPP.fun, POTevents.fun

Examples

```r
data(BarTxTn)

BarEv <- POTevents.fun(T = BarTxTn$Tx, thres = 318,
date = cbind(BarTxTn$ano, BarTxTn$mes, BarTxTn$dia))

# empirical rate based on overlapping intervals
emplambdaB <- emplambdaD.fun(posE = BarEv$Px, inddat = BarEv$inddat, t = c(1:8415),
  lint = 153, tit = "Barcelona")
```

Description

This function calculates the empirical occurrence rates of a point process using disjoint intervals. The rate is assigned to the mean point of the interval. A plot of the empirical rate over time can be performed optionally.

Usage

```r
emplambdaD.fun(posE, t, lint=NULL, nint = NULL, plotEmp = TRUE, inddat = NULL,
  tit = "", scax = NULL, scay = NULL)```
Arguments

- **posE**
  Numeric vector of the position of the occurrence points of the NHPP (or any point process in time).

- **t**
  Time index of the observation period. The simplest option is 1,...,n with n the length of the period.

- **lint**
  Optional (alternative argument to nint). Length of the intervals used to calculate the rates.

- **nint**
  Optional (alternative argument to lint). Number of intervals (of equal length) used to to calculate the rates. It is an alternative way to lint for identifying the intervals.

- **plotEmp**
  Logical flag. If it is TRUE, a plot of the empirical rate is carried out.

- **inddat**
  Optional. Index vector equal to 1 for the observations used in the estimation process. By default, all the observations are considered, see `POTevents.fun`.

- **tit**
  Character string. A title for the plot.

- **scax**
  Optional. A two element vector indicating the x-scale for the plot.

- **scay**
  Optional. A two element vector indicating the y-scale for the plot.

Details

The intervals can be specified either by nint or lint; only one of the arguments must be provided.

Value

A list with elements

- **emplambda**
  Vector of the empirical rates.

- **lint**
  Input argument.

- **nint**
  Input argument.

See Also

- `emplambda.fun`, `fitPP.fun`, `POTevents.fun`

Examples

```r
data(BarTxTn)

BarEv<-POTevents.fun(T=BarTxTn$Tx,thres=318,
date=cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$dia))

# empirical rate based on disjoint intervals using nint to specify the intervals
emplambdaDB<-emplambdaD.fun(posE=BarEv$Px,inddat=BarEv$inddat, t=c(1:8415), nint=55)

# empirical rate based on disjoint intervals using lint to specify the intervals
emplambdaDB<-emplambdaD.fun(posE=BarEv$Px,inddat=BarEv$inddat, t=c(1:8415), lint=153)
```
Method mle for Function extractAIC

Description
Method for generic function extractAIC for objects of the S4-class mle or mlePP. It is the same method as in stats4 (that method is not available outside that package).

Methods

signature(fit = "ANY")
signature(fit = "mle")

fitPP.fun

Fit a non homogeneous Poisson Process

Description
This function fits by maximum likelihood a NHPP where the intensity $\lambda(t)$ is formulated as a function of covariates. It also calculates and plots approximate confidence intervals for $\lambda(t)$.

Usage

fitPP.fun(covariates = NULL, start, fixed=list(), posE = NULL, inddat = NULL, POTob = NULL, nobS = NULL, tind = TRUE, tim = NULL, minfun="nlminb", modCI = "TRUE", CIt = "Transf", clByl = 0.95, tit = "", modSim = "FALSE", dplot = TRUE, xlegend = "topleft", lambdaxlim=NULL,lambdaylim=NULL,...)

Arguments

covariates
  Matrix of the covariates to be included in the linear predictor of the PP intensity (each column is a covariate). It is advisable to give names to the columns of this matrix (using dimnames), since they will be used in the output. Otherwise the default names 'Covariate i' are used. The offset covariates must be included in this matrix. A maximum of 50 covariates are allowed.

start
  Named list of the initial values for the estimation of the $\beta$ parameters (including fixed parameters). The names of the list must be (compulsory): b0 (for the intercept), b1 (for the first column in covariates), b2 (for the second column), b3 (for the third column), etc.

fixed
  Named list of the fixed $\beta$ parameters. The elements of this list must be elements of the list start.

posE
  Optional (see Details section). Numeric vector of the position of the PP occurrence points.
**Details**

A Poisson process (PP) is usually specified by a vector containing the occurrence points of the process $(t_i)_{i=1}^n$ (argument posE). Since PP are often used in the framework of POT models, `fitPP.fun` also provides the possibility of using as input the series of the observed values in a POT model $(x_i)_{i=1}^n$ and the threshold used to define the extreme events (argument POTob).

In the case of PP defined by a POT approach, the observations of the extreme events which are not defined as the occurrence point are not considered in the estimation. This is done through the argument inddat, see `POTevents.fun`. If the input is provided via argument POTob, index inddat is calculated automatically. See Coles (2001) for more details on the POT approach.

The maximization of the loglikelihood function can be done using two different optimization routines, `optim` or `nlminb`, selected in the argument minfun. Depending on the covariates included in the function, one routine can succeed to converge when the other fails.
This function allows us to keep fixed some $\beta$ parameters (offset terms). This can be used to specify an a priori known component to be included in the linear predictor during fitting. The fixed parameters must be specified in the `fixed` argument (and also in `start`); the fixed covariates must be included as columns of `covariates`.

The estimation of the $\hat{\beta}$ covariance matrix is based on the asymptotic distribution of the MLE $\hat{\beta}$, and calculated as the inverse of the negative of the hessian matrix. Confidence intervals for $\lambda(t)$ can be calculated using two approaches specified in the argument `city`. See Casella (2002) for more details on ML theory and delta method.

**Value**

An object of class `mlePP`, which is a subclass of `mle`. Consequently, many of the generic functions with `mle` methods, such as `logLik` or `summary`, can be applied to the output of this function. Some other generic functions related to fitted models, such as AIC or BIC, can also be applied to `mlePP` objects.

**Note**

A homogeneous Poisson process (HPP) can be fitted as a particular case, using an intensity defined by only an intercept and no covariate.

**References**


**See Also**

`POTevents.fun, globalval.fun, VARbeta.fun, CItran.fun, CIdelta.fun`

**Examples**

```r
#model fitted using as input posE and inddat and no confidence intervals
data(BarTxTn)
covB<-cbind(cos(2*pi*BarTxTn$dia/365), sin(2*pi*BarTxTn$dia/365),
BarTxTn$TTx,BarTxTn$Txm31,BarTxTn$Txm31**2)
BarEv<-POTevents.fun(T=BarTxTn$Tx,thres=318,
date=cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$dia))

mod1B<-fitPP.fun(covariates=covB,
posE=BarEv$Px, inddat=BarEv$inddat,
tit="BAR Tx; cos, sin, TTx, T Xm31, T Xm31**2",
start=list(b0=-100,b1=1,b2=-1,b3=0,b4=0,b5=0))
```
GenEnv.fun

Calculation of simulated envelopes

Description

This function calculates a point estimation and an envelope for a given statistic using a Monte Carlo approach. The statistic must be a function of the occurrence points of a NHPP.

It calls the auxiliary function funSim.fun (not intended for the users), see Details section.

Usage

GenEnv.fun(nsim, lambda, fun.name, fun.args = NULL, clevel = 0.95, cores = 1, fixed.seed=NULL)

Arguments

nsim Number of simulations for the calculations.
lambda Numeric vector of the intensity $\lambda(t)$ (or $\hat{\lambda}(t)$) of the NHPP.
fun.name Name of the function defining the statistic to be estimated.
fun.args Additional arguments for the function fun.name.
clevel Confidence level of the envelope.
cores Optional. Number of cores of the computer to be used in the calculations. Default: one core is used.
fixed.seed An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
The auxiliary function funSim.fun generates a simulated sample of the occurrence points in a NHPP and calculates the corresponding statistic using the simulated points.

Value

A list with elements

- valmed: Point estimation (mean value) of the statistic to be calculated.
- valinf: Lower value of the simulated CI.
- valsup: Upper value of the simulated CI.
- lambda: Input argument.
- nsim: Input argument.
- nsimval: Number of valid simulations (used in the calculation of the CI and the point estimation).
- fixed.seed: Input argument.

See Also

- simNHPP.fun, resQQplot.fun

Examples

# Calculation of the point estimation and a 95% CI based on 100 simulations
# for the second occurrence time of a NHPP with intensity lambda.fun.
# posk.fun(x, k) is a function that returns the value in the row k of vector x.

lambda.fun<-runif(1000, 0.01, 0.02)
aux<-GenEnv.fun(lambda=0.1, fun.name="posk.fun", fun.args=2, nsim=100)

# if we want reproducible results, we can fixed the seed in the generation process
# (the number of cores used in the calculations must also be the same to reproduce
# the result)

aux<-GenEnv.fun(lambda=0.1, fun.name="posk.fun", fun.args=2, nsim=100, fixed.seed=123)

# the result (with 1 core): Lower interval: 25.55; Mean value: 136.06; Upper interval: 288

This function performs a thorough validation analysis for a fitted NHPP. It calculates the (generalized) uniform and the raw (or scaled) residuals, performs residual plots for the uniform residuals, and time residual and lurking variable plots for the raw or scaled residuals. It also plots the fitted and empirical estimations of the NHPP intensity. Optionally, it also performs a residual QQplot.
Usage

globalval.fun(mlePP, lint = NULL, nint = NULL, Xvar = NULL, namXvar = NULL, Xvart = NULL, namXvart = NULL, h = NULL, typeres = NULL, typeresLV = "Pearson", typeI = "Disjoint", nsim = 100, clevel = 0.95, resqqplot = FALSE, nintLP = 100, tit = "", flow = 0.5, addlow = FALSE, histWgraph = TRUE, plotDisp = c(2, 2), indgraph = FALSE, scax = NULL, scay = NULL, legcex = 0.5, cores = 1, xlegend = "topleft", fixed.seed = NULL)

Arguments

mlePP An object of class mlePP-class; usually, the output from fitPP.fun.
lint Length of the intervals used to calculate the residuals.
nint Number of intervals used to calculate the residuals. Intervals of equal length are considered. Only used if typeI="Disjoint". In that case, only one of the arguments lint or nint must be specified.
Xvar Optional. Matrix of the lurking variables (each column is a variable).
namXvar Optional. Vector of names of the variables in Xvar.
Xvart Optional. Matrix of the variables for the residual plots (each column is a variable). A time plot is performed in all the cases.
namXvart Optional. Vector of names of the variables in Xvart.
h Optional. Weight function to calculate the scaled residuals. By default, Pearson residuals with
\[ h(t) = 1/\sqrt{\hat{\lambda}(t)} \]
are calculated. This function is used to calculate both the scaled residuals and the residuals for the lurking variables (except if typeResLV="Raw").
typeres Optional. Label indicating the type of scaled residuals. By default, Pearson residuals are calculated and label is "Pearson".
typeresLV Label indicating the type of residuals ("Raw" or any type of scaled residuals such as "Pearson") to calculate the residuals for the lurking variable plots.
typeI Label indicating the type ("Overlapping" or "Disjoint") of intervals used to calculate the residuals.
clevel Confidence level of the residual envelopes.
resqqplot Logical flag. It is is TRUE, a residual qqplot is carried out.
nsim Number of simulations for the residual qqplot.
nintLP Number of levels considered in the lurking variables. It is used as argument nint in the call of the function graphResCov.fun.
tit Character string. A title for the plot.
flow Argument f for the lowess smoother of the raw (or scaled) residual plots, see lowess.
addlow Logical flag. If it is TRUE, a lowess is added in the residual plots.
histgraph Logical flag. If it is TRUE, a new graphical device is opened with the option record=TRUE, so that the history of all plots is recorded in the new device. This option may not work on some platforms; for example, RStudio does not allow the user to open new graphical devices.

plotDisp A vector of the form c(nr, nc). The residual plots in graphresU.fun, graphres.fun and graphResCov.fun will be drawn in a nr×nc layout. It is used as argument mfrow in par. By default, a 2 × 2 layout is used.

indgraph Logical flag. If it is TRUE, the validation plots (except the residual versus variables plots) in graphresU.fun are carried out in four 1 × 1 layouts. By default, a 2 × 2 layout is used.

scax Optional. Vector of two values indicating the range of values for the x-axis in the fitted and empirical rate plot. An adequate range is selected by default.

scay Optional. Vector of two values indicating the range of values for the x-axis in the fitted and empirical rate plot. An adequate range is selected by default.

legcex cex argument for the legend in the residual time plots (see par for details).

cores Optional. Number of cores of the computer to be used in the calculations. Default: one core is used.

xlegend Argument xlegend used in the call of the function graphrate.fun; see that function for details.

fixed.seed An integer or NULL. It is the argument for resQQplot.fun.

Details

If typeI="Overlapping", argument lint is compulsory. If typeI="Disjoint", only one of the arguments lint or nlint must be specified.

Value

A list with the same elements that CalcRes.fun or CalcResD.fun (depending on the value of the argument typeI).

References


See Also
graphres.fun, graphrate.fun, resQQplot.fun, graphResCov.fun, graphresU.fun

Examples

data(BarTxTn)
covB<-cbind(cos(2*pi*BarTxTn$dia/365), sin(2*pi*BarTxTn$dia/365), BarTxTn$TTx,BarTxTn$Txm31,BarTxTn$Txm31**2)
modB<-fitPP.fun(tind=TRUE,covariates=covB, 
POTob=list(T=BarTxTn$Tx, thres=318), 
tit="BAR Tx; cos, sin, TTx, Txm3l, Txm3l**2", 
start=list(b0=-100,b1=1,b2=10,b3=0,b4=0,b5=0),CIty="Transf",modCI=TRUE, 
modSim=TRUE,dplot=FALSE)

#Since only one graphical device is opened and the argument histWgraph is TRUE 
#by default, the different plots can be scrolled up and down with the "Page Up" 
#and "Page Down" keys.

aux<-globalval.fun(mlePP=modB,lint=153,typeI="Disjoint", 
typeRes="Raw",typeResLV="Raw",resqqplot=FALSE)

#if typeRes and typeResLV are not specified, Pearson residuals are calculated
#by default.

aux<-globalval.fun(mlePP=modB,lint=153,typeI="Disjoint", 
resqqplot=FALSE)

---

**graphrate.fun**

*Plot fitted and empirical PP occurrence rates*

**Description**

This function calculates the empirical and the cumulative fitted occurrence rate of a PP on overlapping or disjoint intervals and plot them versus time.

**Usage**

```r
graphrate.fun(objres = NULL, fittedlambda = NULL, emplambda = NULL, t = NULL, 
lint = NULL, typeI = "Disjoint", tit = "", scax = NULL, scay = NULL, 
xlegend = "topleft",histWgraph=TRUE)
```

**Arguments**

- **objres** Optional. A list with (at least) elements fittedlambda, emplambda, t, and typeI. For example, the output from `CalcRes.fun` or `CalcResD.fun`; see those functions for details.
- **fittedlambda** Optional. Numeric vector of the cumulative fitted intensities $\hat{\lambda}(t)$ over the considered intervals (and usually divided by the length of the interval).
- **emplambda** Optional. Numeric vector of the empirical PP occurrence rates estimated over the considered intervals (usually divided by the length of the interval).
- **t** Optional. Time vector of the PP observation period.
- **lint** Optional. Length of the intervals used to calculate the empirical and the (cumulative) fitted occurrence intensities.
typeI  
Label indicating the type (‘Overlapping’ or ‘Disjoint’) of the intervals.

tit  
Character string. A title for the plot.

scax  
Optional. Vector of two values giving the range of values for the x-axis. An adequate range is selected by default.

scay  
Optional. Vector of two values giving the range of values for the y-axis. An adequate range is selected by default.

xlegend  
Label indicating the position where the legend on the graph will be located.

histWgraph  
Logical flag. If it is TRUE, a new graphical device is opened with the option record=True. This option may not work on some platforms; for example, RStudio does not allow the user to open new graphical devices.

Details

Either the argument objres or the set of arguments (fittedlambda, emplambda, t) must be specified. If objres is provided, fittedlambda, emplambda, t, lint and typeI are ignored.

In order to make comparable the empirical and the fitted occurrence rates, a cumulative fitted rate must be used. That means that argument fittedlambda must be the sum of the intensities fitted by the model over the same interval where the empirical rates have been calculated.

See Also

CalcRes.fun, CalcResD.fun

Examples

```r
# plot of rates based on overlapping intervals
graphrate.fun(emplambda=runif(500,0,1), fittedlambda=runif(500,0,1),
t=c(1:500), lint=100, tit="Example", typeI="Overlapping")

# plot of rates based on disjoint intervals
graphrate.fun(emplambda=runif(50,0,1), fittedlambda=runif(50,0,1),
t=c(1:50), lint=10, tit="Example", typeI="Disjoint")

# Example using objres as input. In this example X1 has no influence on the rate;
# consequently the fitted rate is almost a constant.
X1<-rnorm(1000)
modE<-fitPP.fun(tind=TRUE,covariates=cbind(X1),
pose=round(runif(40,1,1000)), inddat=rep(1,1000),
tim=c(1:1000), tit="Simulated example", start=list(b0=1,b1=0),
modCI=FALSE,modSim=TRUE,dplot=FALSE)
ResDE<-CalcResD.fun(mlePP=modE,lint=50)

graphrate.fun(ResDE, tit="Example")
```
**graphres.fun**

*Plot NHPP residuals versus time or monotonous variables*

**Description**

This function plots residuals of a NHPP (raw or scaled, overlapping or disjoint) versus time or other variables which are monotonous functions.

**Usage**

```r
graphres.fun(objres = NULL, typeRes = "Raw", t = NULL, res = NULL, lint = NULL,
posE = NULL, fittedlambda = NULL, typeI = "Disjoint", xvariables = NULL,
namXv = NULL, histWgraph=TRUE, plotDisp=c(2,2), addlow = FALSE, lwd = 2,
tit = "", flow = 0.5, xlegend = "topleft", legcex = 0.5)
```

**Arguments**

- `objres` Optional. A list with (at least) elements `t`, `typeI` and `Rawres` and/or `ScaRes`, depending on the value of `typeRes`. For example, the output list from the functions `CalcRes.fun` or `CalcResD.fun`; see those functions for details.
- `typeRes` Label indicating the type of residuals ("Raw" or any type of scaled residuals such as "Pearson").
- `t` Optional. Time vector of the PP observation period.
- `res` Optional. Vector of residuals.
- `lint` Optional. Length of the intervals used to calculate the residuals.
- `posE` Optional. Numeric vector of the PP occurrence times. Only used when `typeI` = "Overlapping".
- `fittedlambda` Optional. Vector of the cumulative fitted PP intensity over the intervals. Used to calculate the envelopes when `typeRes`="Raw".
- `typeI` Label indicating the type ("Overlapping" or "Disjoint") of intervals.
- `xvariables` Optional. Matrix of the variables for the residual plots (each column is a variable).
- `namXv` Optional. Vector of the names of the variables in `xvariables`.
- `histWgraph` Logical flag. If it is TRUE, a new graphical device is opened with the option `record=TRUE`, so that the history of all plots is recorded in the new device. This option may not work on some platforms; for example, RStudio does not allow the user to open new graphical devices.
- `plotDisp` A vector of the form `c(nr, nc)`. The residual plots will be drawn in a `nr×nc` array. It is used as argument `mrow` in `par`. By default, a `2 × 2` window is used.
- `tit` Character string. A title for the plots.
- `addlow` Logical flag. If it is TRUE, a lowess is added to the residual plots.
- `lwd` Argument `lwd` for plotting the lowess lines, see `par` for details.
- `flow` Argument `f` for the lowess, see `lowess` for details.
- `xlegend` Label giving the position of the graph where the legend will be located.
- `legcex` Argument `cex` for the legend, see `par` for details.
Details

Either argument objres or pair of arguments (t,res) must be specified. If objres is provided, arguments t,res, typeRes, typeI, posE and fittedlambda are ignored.

A residual plot versus time is always performed. These plots are intended for time or variables which are monotonous functions, since residuals are calculated over a given time interval and plotted versus the value of the variables in the mean point of the interval.

A smoother (lowess) of the residuals can be optionally added to the plots. In the case of overlapping intervals, the residuals of the occurrence points are marked differently from the rest. In the case typeRes="Raw" (if argument fittedlambda is available) or typeRes="Pearson", envelopes for the residuals are also plotted. The envelopes are based on an approach analogous to the one shown in Baddeley et al. (2005) for spatial Poisson processes. The envelopes for raw residuals are,

\[ ± \frac{2}{l_2 - l_1} \sqrt{\sum_{i \in (l_1, l_2)} \hat{\lambda}(i)} \]

where index i runs over the integers in the interval \((l_1, l_2)\). The envelopes for the Pearson residuals are,

\[ ± \frac{2}{\sqrt{l_2 - l_1}} \]

These plots allow us to analyze the effect on the intensity, of the covariates included in the model or other potentially influent variables. They show if the mean or the dispersion of the residuals vary sistematically, see for example residual analysis in Atkinson (1985) or Collett (1994).

References


See Also

graphrate.fun

Examples

#Example using objres as input

```r
X1 <- c(1:1000) * 0.5

modE <- fitPP.fun(tind=TRUE, covariates=cbind(X1),
  posE=round(runif(40,1,1000)), inddat=rep(1,1000),
  tim=c(1:1000), tit="Simulated example", start=list(b0=1,b1=0),
  modSim = TRUE, dplot = FALSE)
```
Perform lurking variable plots for a set of variables

Description

This function performs lurking variable plots for a set of variables. The function `graphResX.fun` performs the lurking variable plot for one variable and `graphResCov.fun` calls this function for a set of variables; see `graphResX.fun` for details.

Usage

```r
graphResCov.fun(Xvar, nint, mlePP, h = NULL, typeRes = "Pearson", namX = NULL, histWgraph=TRUE, plotDisp=c(2,2), tit = "")
```

Arguments

- `Xvar`  
  Matrix of variables (each column is a variable).
- `nint`  
  Number of intervals each covariate is divided into to perform the lurking variable plot.
- `mlePP`  
  An object of class `mlePP-class`; usually, the output from `fitPP.fun`.
- `typeRes`  
  Label indicating the type of residuals ("Raw" or any type of scaled residuals such as "Pearson") used in the plots.
- `h`  
  Optional. Weight function used to calculate the scaled residuals (if `typeRes` is not equal to "Raw"). By default, Pearson residuals with \( h(t) = 1/\sqrt{\hat{\lambda}(t)} \) are calculated. \( \hat{\lambda}(t) \) is provided by element lambdafit in mlePP.
- `namX`  
  Optional. Vector of the names of the variables in `Xvar`. 

**histWgraph** Logical flag. If it is TRUE, a new graphical device is opened with the option `record=TRUE`, so that the history of all plots is recorded in the new device. This option may not work on some platforms; for example, RStudio does not allow the user to open new graphical devices.

**plotDisp** A vector of the form `c(nr, nc)`. The lurking variable plots will be drawn in a `nr×nc` array. It is used as argument `mfrow` in `par`. By default, a `2×2` window is used.

**tit** Character string. A title for the plot.

### Value

A list with elements

- **mxres** Matrix of residuals (each column contains the residuals of a variable).
- **mxm** Matrix of mean values (each column contains the mean values of a variable in each interval).
- **mxpc** Matrix of the quantiles that define the intervals of each variable (each column contains the quantiles of one variable).
- **nint** Input argument.
- **mlePP** Input argument.

### References


### See Also

`graphResX.fun`, `graphres.fun`

### Examples

```r
#Simulated process without any relationship with variables Y1 and Y2
#The plots are performed dividing the variables into 50 intervals
#Raw residuals.

Y1<-rnorm(500)
X2<-rnorm(500)
auxmlePP<-fitPP.fun(posE=round(runif(50,1,500)), inddat=rep(1,500),
covariates=cbind(X1,X2), start=list(b0=1,b1=0,b2=0))

Y1<-rnorm(500)
```
Y2<-rnorm(500)
res<graphResCov.fun(mlePP=auxmlePP, Xvar=cbind(Y1,Y2), nint=50,
typeRes="Raw",namX=c("Y1","Y2"),plotDisp=c(2,1))

#If more variables were specified in the argument Xvar, with
#the same 2X1 layout specified in plotDisp, the resulting plots could be
#scrolled up and down with the "Page Up" and "Page Down" keys.

---

**graphresU.fun**  
*Validation analysis of PP uniform (generalized) residuals*

**Description**

This function checks the properties that must be fulfilled by the uniform (generalized) residuals of a PP: uniform character and uncorrelation. Optionally, the existence of patterns versus covariates or potentially influent variables can be graphically analyzed.

**Usage**

```r
graphresU.fun(unires, posE, Xvariables = NULL, namXv = NULL, flow = 0.5,
tit = "", addlow = TRUE, histWgraph=TRUE, plotDisp=c(2,2), indgraph = FALSE)
```

**Arguments**

- `unires`  
  Numeric vector of the uniform residuals.

- `posE`  
  Numeric vector of the occurrence times of the PP.

- `Xvariables`  
  Matrix of variables to perform the residual plots (each column is a variable).

- `namXv`  
  Optional. Vector of names of the variables in Xvariables.

- `tit`  
  Character string. A title for the plot.

- `addlow`  
  Logical flag. If it is TRUE, a lowess is added to the plots.

- `flow`  
  Argument f for the lowess smoother; see `lowess` for details.

- `histWgraph`  
  Logical flag. If it is TRUE, a new graphical device is opened with the option record=TRUE, so that the history of all plots is recorded in the new device. This option may not work on some platforms; for example, RStudio does not allow the user to open new graphical devices.

- `plotDisp`  
  A vector of the form c(nr, nc). The residual versus variables plots will be drawn in a nr×nc array. It is used as argument mrow in `par`. By default, a 2 × 2 layout is used.

- `indgraph`  
  Logical flag. If it is TRUE, the validation plots (except the residuals versus variables plots) are carried out in four1 × 1 layouts. By default, a 2 × 2 layout is used.
The validation analysis of the uniform character consists in a uniform Kolmogorov-Smirnov test and a qqplot with a 95% confidence band based on a beta distribution. The analysis of the serial correlation is based on the Pearson correlation coefficient, Ljung-Box tests and a lagged serial correlation plot. An index plot of the residuals and residual plots versus the variables in argument Xvariables are performed to analyze the effect of covariates or other potentially influent variables. These plots will show if the mean or dispersion of the residuals vary sistematically, see model diagnostic of Cox-Snell residuals in Collett (1994) for more details.

References


See Also

unifres.fun, transfH.fun

Examples

```r
#Since only one graphical device is opened and the argument histWgraph
#is TRUE by default, the resulting residual plots (three pages with the
#considered 1X2 layout for the residual versus variables plot)
#can be scrolled up and down with the "Page Up" and "Page Down" keys.

X1<-rnorm(500)
X2<-rnorm(500)

graphresU.fun(unires=runif(30,0,1),posE=round(runif(30,0,500)),
Xvariables=cbind(X1,X2), namXv=c("X1","X2"),tit="Example",flow=0.7,plotDisp=c(1,2))
```

Description

This function performs a lurking variable plot to analyze the residuals in terms of different levels of the variable.
Usage

```r
graphResX.fun(X, nint, mlePP, typeRes = "Pearson", h = NULL, namX = NULL)
```

Arguments

- **X**: Numeric vector, the variable for the lurking variable plot.
- **nint**: Number of intervals or levels the variable is divided into.
- **mlePP**: An object of class `mlePP-class`; usually, the output from `fitPP.fun`.
- **typeRes**: Label indicating the type of residuals ("Raw" or any type of scaled residuals such as "Pearson").
- **h**: Optional. Weight function used to calculate the scaled residuals (if `typeRes` is not equal to 'Raw'). By default, Pearson residuals with \( h(t) = 1/\sqrt{\hat{\lambda}(t)} \) are calculated. \( \hat{\lambda}(t) \) is provided by the lambdafit slot in mlePP.
- **namX**: Optional. Name of variable X.

Details

The residuals for different levels of the variable are analyzed. For a variable \( X(t) \), the considered levels are

\[
W(P_{X,j}, P_{X,j+1}) = \{ t : P_{X,j} \leq X(t) < P_{X,j+1} \}
\]

where \( P_{X,i} \) is the sample j-percentile of X. This type of plot is specially useful for variables which are not a monotonous function of time.

In the case `typeRes='Raw'` or `typeRes='Pearson'`, envelopes for the residuals are also plotted. The envelopes are based on an approach analogous to the one in Baddeley et al. (2005) for spatial Poisson processes. The envelopes for raw residuals are

\[
\pm \frac{2}{l_W} \sqrt{\sum_i \hat{\lambda}(i)}
\]

where index \( i \) runs over the integers in the level \( W(P_{X,j}, P_{X,j+1}) \), and \( l_W \) is its length (number of observations in \( W \)). The envelopes for the Pearson residuals are,

\[
\pm 2/\sqrt{l_W}.
\]

Value

A list with elements

- **xres**: Vector of residuals.
- **xm**: Vector of the mean value of the variable in each interval.
- **pc**: Vector of the quantiles that define the levels of the variable.
- **typeRes**: Input argument.
- **namX**: Input argument.
- **lambdafit**: Input argument.
- **pose**: Input argument.
References


See Also

`graphResCov.fun`, `graphres.fun`

Examples

```r
##Simulated process not related to variable X
##Plots dividing the variable into 50 levels
X1<-rnorm(500)
X2<-rnorm(500)
auxmlePP<-fitPP.fun(posE=round(runif(50,1,500)), inndat=rep(1,500),
covariates=cbind(X1,X2),start=list(b0=1,b1=0,b2=0))

##Raw residuals
res<-graphResX.fun(X=rnorm(500),nint=50,mlePP=auxmlePP,typeRes="Raw")

##Pearson residuals
res<-graphResX.fun(X=rnorm(500),nint=50,mlePP=auxmlePP,typeRes="Pearson")
```

---

**LRTpv.fun**

*Calculate the p-value of a likelihood ratio test for each covariate in the model*

###Description

This function calculates, for each covariate in the model (except the intercept), the p-value of a likelihood ratio test comparing the original fitted NHPP with the model excluding that covariate from the linear predictor.

###Usage

`LRTpv.fun(mlePP)`
Arguments

mlePP  An object of class mlePP-class; usually, the output from fitPP.fun. The fitted model cannot include fixed parameters.

Details

A LRT is carried for all the covariates in the linear predictor except the intercept. If the model has not an intercept and there is only one covariate, no test can be carried out.

Value

A matrix with one column, which contains the LRT p-values for all the covariates in the model (except the intercept)

See Also

fitPP.fun, testlik.fun, dropAIC.fun, addAIC.fun

Examples

data(Bartxtn)
covb<-cbind(cos(2*pi*Bartxtn$dia/365), sin(2*pi*Bartxtn$dia/365), Bartxtn$TTx,Bartxtn$Tm31,Bartxtn$Tm31**2)
BarEv<-POTevents.fun(T=Bartxtn$Tx, thres=318, date=cbind(Bartxtn$ano,Bartxtn$mes,Bartxtn$dia))

mod1B<-fitPP.fun(tind=TRUE, covariates=covb, posE=BarEv$Px, inddat=BarEv$inddat, tit="BAR Tx; cos, sin, TTx, Tm31, Tm31**2", start=list(b0=-100,b1=1,b2=10,b3=0,b4=0,b5=0),dplot=FALSE, modCI=FALSE)

LRTpv.fun(mod1B)

mlePP-class  

Class "mlePP" for results of maximum likelihood estimation of Poisson processes with covariates

Description

This class encapsulates the output from the maximum likelihood estimation of a Poisson process where the intensity is modeled as a linear function of covariates.

Objects from the Class

Objects can be created by calls of the form new("mlePP", ...), but most often as the result of a call to fitPP.fun.
Slots

call: Object of class "language". The call to `fitPP.fun`.

coeff: Object of class "numeric". The estimated coefficients of the model.

fullcoeff: Object of class "numeric". The full coefficient vector, including the fixed parameters of the model. It has an attribute, called 'TypeCoeff' which shows the names of the fixed parameters.

vcov: Object of class "matrix". Approximate variance-covariance matrix of the estimated coefficients. It has an attribute, called 'CalMethod' which shows the method used to calculate the inverse of the information matrix: 'Solve function', 'Cholesky', 'Not possible' or 'Not required' if modCI=FALSE.

min: Object of class "numeric". Minimum value of objective function, that is the negative of the loglikelihood function.

details: Object of class "list". The output returned from `optim`. If `nlminb` is used to minimize the function, it is NULL.

minuslogl: Object of class "function". The negative of the loglikelihood function.

nobs: Object of class "integer". The number of observations.

method: Object of class "character". It is a bit different from the slot in the extended class `mle` here, it is the input argument `minfun` of `fitPP.fun` instead of the method used in `optim` (this information already appears in details).

detailsb: Object of class "list". The output returned from `nlminb`. If `optim` is used to minimize the function, it is NULL.

npar: Object of class "integer". Number of estimated parameters.

inddat: Object of class "numeric". Input argument of `fitPP.fun`.

lambdafit: Object of class "numeric". Vector of the fitted intensity \( \hat{\lambda}(t) \).

llambda: Object of class "numeric". Vector of lower limits of the CI.

ulambda: Object of class "numeric". Vector of upper limits of the CI.

convergence: Object of class "integer". A code of convergence. 0 indicates successful convergence.

posE: Object of class "numeric". Input argument of `fitPP.fun`.

covariates: Object of class "matrix". Input argument of `fitPP.fun`.

fixed: Object of class "list". Input argument of `fitPP.fun`.

tit: Object of class "character". Input argument of `fitPP.fun`.

tind: Object of class "logical". Input argument of `fitPP.fun`.

t: Object of class "numeric". Input argument of `fitPP.fun`.

Extends

Class "mle", directly.
Methods

Most of the S4 methods in \texttt{stats4} for the S4-class \texttt{mle} can be used. Also a \texttt{mle} method for the generic function \texttt{extractAIC} and a version of the \texttt{profile mle} method adapted to the \texttt{mlePP} objects are available:

- \texttt{coef} \ signatures(object = "mle")
- \texttt{logLik} \ signatures(object = "mle")
- \texttt{nobs} \ signatures(object = "mle")
- \texttt{show} \ signatures(object = "mle")
- \texttt{summary} \ signatures(object = "mle")
- \texttt{update} \ signatures(object = "mle")
- \texttt{vcov} \ signatures(object = "mle")
- \texttt{confint} \ signatures(object = "mle")
- \texttt{extractAIC} \ signatures(object = "mle")
- \texttt{profile} \ signatures(fitted = "mlePP")

Some other generic functions related to fitted models, such as \texttt{AIC} or \texttt{BIC}, can also be applied to \texttt{mlePP} objects.

Note

Let us remind that, as in all the S4-classes, the symbol \texttt{@} must be used instead of $ to name the slots: \texttt{mlePP@covariates}, \texttt{mlepp@lambdafit}, etc.

See Also

- \texttt{fitPP.fun}, \texttt{mle}

Examples

\begin{verbatim}
showClass("mlePP")
\end{verbatim}

\begin{verbatim}
POTevents.fun
\end{verbatim}

\textit{Calculate extreme events using a POT approach}

Description

This function calculates the characteristics of the extreme events of a series \((x_i)\) defined using a peak over threshold (POT) method with an extreme threshold. The initial and the maximum intensity positions, the mean excess, the maximum excess and the length of each event are calculated.

Usage

\begin{verbatim}
POTevents.fun(T, thres, date = NULL)
\end{verbatim}
Arguments

T Numeric vector, the series \( (x_i) \) to calculate the extreme events.

thres Threshold value used to define the extreme events.

date Optional. A vector or matrix indicating the date of each observation.

Details

One of the elements of the output from this function is a vector (inddat) which marks the observations that should be used in the estimation of a point process, resulting from a POT approach. The observations to be considered in the estimation are marked with 1 and correspond to the non occurrence observations and to a single occurrence point per event. The occurrence point is defined as the point where maximum intensity of the event occurs. The observations in an extreme event which are not the occurrence point are marked with 0 and treated as non observed.

Value

A list with components

Pi Vector of the initial points of the extreme events.

datePi Date of the initial points Pi.

Px Vector of the points of maximum excess of the extreme events.

datePx Vector of the date of the maximum excess points Px.

Im Vector of the mean excesses (over the threshold) of the extreme events.

Ix Vector of the maximum excesses (over the threshold) of the extreme events.

L Vector of the lengths of the extreme events.

inddat Index equal to 1 in the observations used in the estimation process and to 0 in the others.

See Also

fitPP.fun

Examples

data(BarTxTn)
dateB<-cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$diames)
BarEv<-POTevents.fun(T=BarTxTn$Tx,thres=318, date=dateB)
### Profile Methods

**Method mlePP for Function profile**

### Description

Method for generic function `profile` for objects of the S4-class `mlePP`. It is almost identical to the method `mle` for this function in `stats4`, but small changes have to be done due to the differences in the arguments of the functions `mle` and `fitPP.fun`. In order to profile an `mlePP` object, its `vcov` slot cannot be missing. That means that if the function `fitPP.fun` is used to create the object, the argument `modCI=TRUE` must be used.

### Methods

signature(fitted = "mlePP")

### resQQplot.fun

**Perform a qqplot for the residuals of a NHPP**

### Description

This function performs a qqplot comparing the empirical quantiles of the residuals with the expected quantiles under the fitted NHPP, calculated by a Monte Carlo approach.

It calls the auxiliary function `resSim.fun` (not intended for the users), see Details section.

### Usage

```r
resQQplot.fun(nsim, objres, covariates, clevel = 0.95, cores = 1, tit = "", fixed.seed=NULL, histWgraph=TRUE)
```

### Arguments

- **nsim**: Number of simulations for the calculations.
- **objres**: A list with the same elements of the output list from the function `CalcRes.fun` or `CalcResD.fun`.
- **covariates**: Matrix of covariates to fit the NHPP (each column is a covariate).
- **clevel**: Confidence level of the residual envelope.
- **cores**: Optional. Number of cores of the computer to be used in the calculations. Default: one core is used.
- **tit**: Character string. A title for the plot.
- **fixed.seed**: An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.
- **histWgraph**: Logical flag. Only used in Windows platforms. If it is TRUE, a new graphical device is opened with the option `record=TRUE`. 
Details

The expected quantiles are calculated as the median values of the simulated samples. Confidence intervals for each quantile \( r^{(i)} \) with pointwise significance level \( clevel \) are calculated as quantiles of probability \( 1 - clevel / 2 \) and \( clevel / 2 \) of the simulated sample for each residual.

All type of residuals (disjoint or overlapping and Pearson or raw residuals) are supported by this function. However, the qqplot for overlapping residuals can be a high time consuming process. So, disjoint residuals should be preferred in this function.

The auxiliary function `resSim.fun` generates a NHPP with intensity \( \lambda(t) \), fits the model using the covariate matrix and calculates the residuals.

Value

A list with elements

- `resmed`: Numeric vector containing the mean of the simulated residuals in each point.
- `ressup`: Numeric vector of the upper values of the simulated envelopes.
- `resinf`: Numeric vector of the lower values of the simulated envelopes.
- `objres`: Input argument.
- `nsim`: Input argument.
- `fixed.seed`: Input argument.

References


See Also

`simNHP.fun, GenEnv.fun`

Examples

```r
X1 <- rnorm(500)
X2 <- rnorm(500)

aux <- fitPP.fun(tind=TRUE, covariates=cbind(X1, X2),
                 posE=round(runif(40, 1, 500)), inddat=rep(1, 500),
                 tim=c(1:500), tit="Simulated example", start=list(b0=1, b1=0, b2=0), dplot=FALSE)

auxRes <- CalcRes0.fun(mlePP=aux, lint=50)

# If we want reproducible results, we can fixed the seed in the generation process
# (the number of cores used in the calculations must also be the same to reproduce the result)

auxqq <- resQQplot.fun(nsim=50, objres=auxRes, covariates=cbind(X1, X2), fixed.seed=123)
```
**Description**

This function generates the occurrence times of the points of a NHPP with a given time-varying intensity $\lambda(t)$, in a period $(0, T)$. The length of argument lambda determines $T$, the length of the observation period.

It calls the auxiliary function buscar (not intended for the users), see Details section.

**Usage**

```r
simNHP.fun(lambda, fixed.seed=NULL)
```

**Arguments**

- `lambda`: Numeric vector, the time varying intensity $\lambda(t)$ to generate the NHPP.
- `fixed.seed`: An integer or NULL. If it is an integer, that is the value used to set the seed in random generation processes. It it is NULL, a random seed is used.

**Details**

The generation of the NHPP points consists in two steps. First, the points of a homogeneous PP of intensity 1 are generated using independent exponentials. Then, the homogeneous occurrence times are transformed into the points of a non homogeneous process with intensity $\lambda(t)$. This transformation is performed by the auxiliary function buscar (not intended for the user).

**Value**

A list with elements

- `posNH`: Numeric vector of the occurrences times of the NHPP generated in the observation period $(0,T)$.
- `lambda`: Input argument.
- `fixed.seed`: Input argument.

**References**


**See Also**

`GenEnv.fun`, `resQQplot.fun`
Examples

# Generation of the occurrence times of a homogeneous Poisson process with constant intensity # 0.01 in a period of time of length 1000

aux<-simNHP.fun(lambda=rep(0.01,1000))
aux$posNH

# If we want reproducible results, we can fix the seed in the generation process
aux<-simNHP.fun(lambda=rep(0.01,1000),fixed.seed=123)
aux$posNH

# And the result is:
# [1] 85 143 275 279 284 316 347 362 634 637 738 786 814 852 870 955

# Generation of the occurrence times of a NHPP with time-varying intensity t in a period of time of length 500

t<-runif(500, 0.01, 0.1)
aux<-simNHP.fun(lambda=t)
aux$posNH

stepAICmle.fun

Choose the best PP model by AIC in a stepwise algorithm

Description

Performs stepwise model selection by AIC for Poisson process models estimated by maximum likelihood.

It calls the auxiliary function checkdim (not intended for the users).

Usage

stepAICmle.fun(ImlePP, covariatesAdd = NULL, startAdd = NULL, direction = "forward", ...)

Arguments

ImlePP
A mlePP-class object; usually the output from fitPP.fun. It defines the initial model of the stepwise algorithm. The fitted model cannot include fixed parameters.

covariatesAdd
Matrix of the potential covariates to be added to the model; each column must contain a covariate. In the 'forward' and the 'both' directions, it is compulsory to assign a matrix to this argument. It is advisable to give names to the columns of this matrix (using dimnames) since, they will be used in the output. Otherwise the default names 'New Covariate i' are used.

startAdd
Optional. The vector of initial values for the estimation of the coefficients of each potential covariate. If it is NULL, initial values equal to 0 are used.
direction: Label indicating the direction of the algorithm: 'forward' (the default), 'backward' or 'both'.

... Further arguments to pass to addAIC.fun and dropAIC.fun, for example the constant k for the AIC calculation

Details

Three directions, forward, backward and both, are implemented. The initial model is given by ImlePP and the algorithm stops when none of the covariates eliminated from the model or added from the potential covariates set (argument covariatesAdd) improves the model fitted in the previous step, according to the AIC. For the 'both' and 'forward' directions, the argument covariatesAdd is compulsory, and the default NULL leads to an error.

In the 'both' direction, 'forward' and 'backward' steps are carried out alternatively. In the 'forward' direction, the initial model usually contains only the intercept.

Value

A mlePP-class object, the fit of the final PP model selected by the algorithm.

References


See Also

addAIC.fun, dropAIC.fun, testlik.fun

Examples

data(BarTxTn)
BarEv<-POTevents.fun(T=BarTxTn$Tx, thres=318,
date=cbind(BarTxTn$ano,BarTxTn$mes,BarTxTn$dia))

#The initial model contains only the intercept
mod1Bind<-fitPP.fun(covariates=NULL, posE=BarEv$Px, inddat=BarEv$inddat,
tit='BAR Intercept ', start=list(b0=1))
#the potential covariates
covB<-cbind(cos(2*pi*BarTxTn$dia/365), sin(2*pi*BarTxTn$dia/365),
BarTxTn$TTx,BarTxTn$Ttxm31,BarTxTn$Txm31**,2)
dimnames(covB)<-list(NULL,c('cos','sin','TTx','Ttxm31', 'Txm31**2'))

bb<-stepAICmle.fun(ImlePP=mod1Bind, covariates=covB, startAdd=c(1,-1,0,0,0),
direction='both')
**testlik.fun**  
*Likelihood ratio test to compare two nested models*

**Description**

This function performs a likelihood ratio test, a test to compare the fit of two models, where the first one (the null model ModR) is a particular case of the other (the alternative model ModG).

**Usage**

```r
testlik.fun(ModG, ModR)
```

**Arguments**

- **ModG** An object of class `mlePP-class`; usually, the output from `fitPP.fun`
- **ModR** An object of class `mlePP-class`; usually, the output from `fitPP.fun`

**Details**

The test statistic is twice the difference in the log-likelihoods of the models. Under the null, the statistic follows a $\chi^2$ distribution with degrees of freedom df2-df1, the number of parameters of modG and modR respectively.

**Value**

A list with elements

- **pv** P-value of the likelihood ratio test.
- **ModG** Input argument.
- **ModR** Input argument.

**References**


**See Also**

`fitPP.fun`, `LRTpv.fun`
Examples

```r
# The alternative model modB is specified by the output fitPP.fun
# The null model modBR is specified by a list with elements llik and npar

data(Bartxtn)

covB <- cbind(cos(2*pi*Bartxtn$dia)/365), sin(2*pi*Bartxtn$dia)/365),
       Bartxtn$TTx,Bartxtn$Txm31,Bartxtn$Txm31**2)

modB <- fitPP.fun(tind=TRUE, covariates=covB,
                    POTob=list(T=Bartxtn$Tx, thres=318),
                    tim=c(1:8415), tit="BAR Tx; cos, sin, TTx, Txm31, Txm31**2",
                    start=list(b0=-100, b1=1, b2=10, b3=0, b4=0, b5=0),
                    dplot=FALSE, modCI=TRUE, modSim=TRUE)

modBR <- fitPP.fun(tind=TRUE, covariates=covB[,1:4],
                     POTob=list(T=Bartxtn$Tx, thres=318),
                     tim=c(1:8415), tit="BAR Tx; cos, sin, TTx, Txm31",
                     start=list(b0=-100, b1=1, b2=10, b3=0, b4=0),
                     dplot=FALSE, modCI=TRUE, modSim=TRUE)

aux <- testlik.fun(ModG=modB, ModR=modBR)
```

---

**transfH.fun**

*Transform a NHPP into a HPP*

**Description**

This function transforms the points \( t_i^N \) of a NHPP into the occurrence points \( t_i^H \) of a HPP of rate 1.

**Usage**

```r
transfH.fun(mlePP)
```

**Arguments**

- **mlePP**
  
  An object of class `mlePP-class`; usually, the output from `fitPP.fun`
Details

Transformation of the NHPP points $t_{i}^{NH}$ into the HPP points $t_{i}^{H}$ is based on the time scale transformation,

$$ t_{i}^{H} = \int_{0}^{t_{i}^{NH}} \lambda(t) dt. $$

(usually the estimated value $\hat{\lambda}(t)$ is used in the transformation.)

Value

A list with elements

- `posE` Numeric vector of the transformed occurrence times of the HPP.
- `lambdafit` Slot of the input argument mlePP.
- `inddat` Slot of the input argument mlePP.

References


See Also

`simNHP.fun`

Examples

```r
X1<-rnorm(500)
X2<-rnorm(500)
auxmlePP<-fitPP.fun(posE=round(runif(50,1,500)), inddat=rep(1,500),
covariates=cbind(X1,X2),start=list(b0=1,b1=0,b2=0))

posEH<-transfH.fun(auxmlePP)
```
unifres.fun 

*Calculate exponential and uniform (generalized) residuals of a HPP*

**Description**

This function calculates the exponential \(d_i\) and the uniform (generalized) residuals \(u_i\) of a HPP, using the occurrence points \(t_i\).

**Usage**

`unifres.fun(posEH)`

**Arguments**

- `posEH` Numeric vector, the occurrence points of a HPP.

**Details**

The exponential residuals of a HPP are defined as the inter-event distances \(d_i = t_i - t_{i-1}\), that are an i.i.d. exponential sample. The series \(d_i\) is an example of the generalized residuals proposed by Cox and Snell (1968). The uniform residuals, defined as the function \(\exp(-d_i)\) of the exponential residuals, are an i.i.d. uniform sample, see Ogata (1988).

**Value**

A list with elements

- `expres` Numeric vector of the exponential residuals.
- `unires` Numeric vector of the uniform residuals.
- `posEH` Input argument.

**References**


**See Also**

`transfH.fun`, `graphresU.fun`
Examples

```r
## generates the occurrence times of a homogeneous PP with constant intensity 0.01
## and calculates de residuals

aux<-simNHP.fun(lambda=rep(0.01,1000))
res<-unifres.fun(aux$posNH)
```

VARbeta.fun  

*Calculate the covariance matrix of the $\hat{\beta}$ vector.*

Description

This function estimates the covariance matrix of the ML estimators of the $\beta$ parameters, using the asymptotic distribution and properties of the ML estimators.

Usage

`VARbeta.fun(covariates, lambdafit)`

Arguments

- `covariates`  
  Matrix of covariates (each column is a covariate).
- `lambdafit`  
  Numeric vector, the fitted PP intensity $\hat{\lambda}(t)$.

Details

The covariance matrix is calculated as the inverse of the negative of the hessian matrix. The inverse of the matrix is calculated using the solve function. If this function leads to an error in the calculation, the inverse is calculated via its Cholesky decomposition. If this option also fails, the covariance matrix is not estimated and a matrix of dimension $0 \times 0$ is returned.

Value

`VARbeta`  

Covariance matrix of the $\hat{\beta}$ vector. It has an attribute, called 'CalMethod' which shows the method used to calculate the inverse of the matrix: 'Solve function', 'Cholesky' or 'Not possible'.

Note

The function `fitPP.fun` calls this function.

References


See Also

CItran.fun, CIDelta.fun

Examples

```r
lambdafit <- runif(100, 0, 1)
X <- cbind(rep(1, 100), rnorm(100), rnorm(100))

aux <- VARbeta.fun(covariates = X, lambdafit = lambdafit)
```
Index

*Topic methods
  extractAIC-methods, 17
  profile-methods, 38
*Topic non homogeneous Poisson process
  NHPoisson-package, 2

addAIC.fun, 3, 14, 34, 42
AIC, 3, 13

BarTtxtn, 4
buscar (simNHP.fun), 40
CalcRes.fun, 5, 9, 23–26, 38
CalcResD.fun, 7, 23–26, 38
checkdim (stepAICmle.fun), 41
CIDelta.fun, 9, 11, 18, 19, 48
CITran.fun, 10, 10, 18, 19, 48
confint, 12
confintAsin.fun, 11
dimnames, 17, 41
dropAIC.fun, 4, 13, 34, 42
eemplambda.fun, 14, 16
eemplambdD.fun, 15, 15
extractAIC, 17, 36
extractAIC, ANY-method
  (extractAIC-methods), 17
extractAIC, mle-method
  (extractAIC-methods), 17
extractAIC-methods, 17

fitPP.fun, 3, 5, 7, 10–13, 15, 16, 17, 18, 22,
  28, 32, 34–37, 41, 43, 44, 47
funSim.fun (GenEnv.fun), 20

GenEnv.fun, 20, 39, 40
globalval.fun, 19, 21
graphrate.fun, 23, 24, 27
graphres.fun, 6, 9, 23, 26, 29, 33
graphResCov.fun, 22, 23, 28, 33
graphresU.fun, 23, 30, 46
graphResX.fun, 28, 29, 31
logLik, 19
lowess, 22, 26, 30
LRTpv.fun, 4, 14, 33, 43
mle, 17, 19, 35, 36, 38
mlePP, 3, 12, 13, 17, 19, 36, 38, 41, 42
mlePP-class, 34
NHPoisson (NHPoisson-package), 2
NHPoisson-package, 2
nlminb, 18, 35
optim, 18, 35
par, 23, 26, 29, 30
pokk.fun (GenEnv.fun), 20
POEvents.fun, 14–16, 18, 19, 36
profile, 36, 38
profile, mlePP-method (profile-methods), 38
profile-methods, 38
resQQplot.fun, 21, 23, 38, 40
resSim.fun (resQQplot.fun), 38
simNHP.fun, 21, 39, 40, 45
stepAICmle.fun, 3, 4, 14, 41
summary, 19
testlik.fun, 34, 42, 43
transfH.fun, 31, 44, 46
unifres.fun, 6, 9, 31, 46
VARbeta.fun, 10–12, 19, 47