Package ‘stelfi’

October 24, 2023

Type Package

Title Hawkes and Log-Gaussian Cox Point Processes Using Template Model Builder

Version 1.0.1

Date 2023-09-29

Depends R (>= 4.3.0)

Imports TMB (>= 1.9.6), sf (>= 1.0.14), fmesher, Matrix, ggplot2 (>= 3.4.3), dplyr (>= 1.1.3), gridExtra (>= 2.3)

LinkingTo TMB, RcppEigen

Description Fit Hawkes and log-Gaussian Cox process models with extensions. Introduced in Hawkes (1971) <doi:10.2307/2334319> a Hawkes process is a self-exciting temporal point process where the occurrence of an event immediately increases the chance of another. We extend this to consider self-inhibiting process and a non-homogeneous background rate. A log-Gaussian Cox process is a Poisson point process where the log-intensity is given by a Gaussian random field. We extend this to a joint likelihood formulation fitting a marked log-Gaussian Cox model. In addition, the package offers functionality to fit self-exciting spatiotemporal point processes. Models are fitted via maximum likelihood using 'TMB' (Template Model Builder). Where included 1) random fields are assumed to be Gaussian and are integrated over using the Laplace approximation and 2) a stochastic partial differential equation model, introduced by Lindgren, Rue, and Lindström. (2011) <doi:10.1111/j.1467-9868.2011.00777.x>, is defined for the field(s).

License GPL (>= 3)

URL https://github.com/cmjt/stelfi/

BugReports https://github.com/cmjt/stelfi/issues

LazyData TRUE

LazyDataCompression xz

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 3.1.10), rmarkdown, parallel, spatstat.utils, spatstat.geom, hawkesbow, covr, knitr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Charlotte M. Jones-Todd [aut, cre, cph]
(<https://orcid.org/0000-0003-1201-2781>, Charlotte Jones-Todd
wrote and continued developmend of the main code.),
Alec van Helsdingen [aut] (Alec van Helsdingen wrote the Hawkes
templates and extended self-exciting TMB templates),
Xiangjie Xue [ctb] (Xiangjie Xue worked the early spatio-temporal
self-exciting TMB templates),
Marsden Fund 3723517 [fnd],
Asian Office of Aerospace Research & Development FA2386-21-1-4028 [fnd]

Maintainer Charlotte M. Jones-Todd <cmjonestodd@gmail.com>

Repository CRAN

Date/Publication 2023-10-24 10:10:02 UTC

R topics documented:

fit_hawkes . . . 3
fit_lgcp . . . 6
fit_mlgcp . . . 9
fit_stelfi . . . 11
get_coefs . . . 14
get_fields . . . 15
get_weights . . . 16
horse_mesh . . . 16
horse_sf . . . 17
iraq_terrorism . . . 17
marked . . . 18
meshmetrics . . . 18
mesh_2_sf . . . 19
multi_hawkes . . . 20
nz_earthquakes . . . 20
nz_murders . . . 21
retweets_niwa . . . 21
sasquatch . . . 22
show_field . . . 23
show_hawkes . . . 24
show_lambda . . . 25
sim_hawkes . . . 26
sim_lgcp . . . 27
stelfi . . . 28
uk_serial . . . 29
xyt . . . 30

Index **31**

fit_hawkes *Self-exciting Hawkes process(es)*

Description

Fit a Hawkes process using Template Model Builder (TMB). The function `fit_hawkes()` fits a self-exciting Hawkes process with a constant background rate. Whereas, `fit_hawkes_cbf()` fits a Hawkes processes with a user defined custom background function (non-homogeneous background rate). The function `fit_mhawkes()` fits a multivariate Hawkes process that allows for between- and within-stream self-excitement.

Usage

```
fit_hawkes(
  times,
  parameters = list(),
  model = 1,
  marks = c(rep(1, length(times))),
  tmb_silent = TRUE,
  optim_silent = TRUE,
  ...
)

fit_hawkes_cbf(
  times,
  parameters = list(),
  model = 1,
  marks = c(rep(1, length(times))),
  background,
  background_integral,
  background_parameters,
  background_min,
  tmb_silent = TRUE,
  optim_silent = TRUE
)

fit_mhawkes(
  times,
  stream,
  parameters = list(),
  tmb_silent = TRUE,
  optim_silent = TRUE,
  ...
)
```

Arguments

`times`	A vector of numeric observed time points.

parameters — A named list of parameter starting values:

- mu, a vector of base rates for each stream of the multivariate Hawkes process,
- alpha, a matrix of the between- and within-stream self-excitement: the diagonal elements represent the within-stream excitement and the off-diagonals the excitement between streams,
- beta, a vector of the exponential intensity decay for each stream of the multivariate Hawkes process,

model — A numeric indicator specifying which model to fit:

- model = 1, fits a Hawkes process with exponential decay (default);
- model = 2, fits a Hawkes process with an alpha that can be negative.

marks — Optional, a vector of numeric marks, defaults to 1 (i.e., no marks).

tmb_silent — Logical, if TRUE (default) then TMB inner optimisation tracing information will be printed.

optim_silent — Logical, if TRUE (default) then for each iteration optim() output will be printed.

... — Additional arguments to pass to optim()

background — A function taking one parameter and an independent variable, returning a scalar.

background_integral — The integral of background.

background_parameters — The parameter(s) for the background function background. This could be a list of multiple values.

background_min — A function taking one parameter and two points, returns min of background between those points.

stream — A character vector specifying the stream ID of each observation in times

Details

A univariate Hawkes (Hawkes, AG. 1971) process is a self-exciting temporal point process with conditional intensity function $\lambda(t) = \mu + \alpha \Sigma_{i:\tau_i<t} e^{(-\beta*(t-\tau_i))}$. Here μ is the constant baseline rate, α is the instantaneous increase in intensity after an event, and β is the exponential decay in intensity. The term $\Sigma_{i:\tau_i<t} \cdots$ describes the historic dependence and the clustering density of the temporal point process, where the τ_i are the events in time occurring prior to time t. From this we can derive the following quantities 1) $\frac{\alpha}{\beta}$ is the branching ratio, it gives the average number of events triggered by an event, and 2) $\frac{1}{\beta}$ gives the rate of decay of the self-excitement. Including mark information results in the conditional intensity $\lambda(t; m(t)) = \mu + \alpha \Sigma_{i:\tau_i<t} m(\tau_i) e^{(-\beta*(t-\tau_i))}$, where $m(t)$ is the temporal mark. This model can be fitted with fit_hawkes().

An in-homogenous marked Hawkes process has conditional intensity function $\lambda(t) = \mu(t) + \alpha \Sigma_{i:\tau_i<t} e^{(-\beta*(t-\tau_i))}$. Here, the background rate, $\mu(t)$, varies in time. Such a model can be fitted using fit_hawkes_cbf() where the parameters of the custom background function are estimated before being passed to TMB.

A multivariate Hawkes process that allows for between- and within-stream self-excitement. The conditional intensity for the j^{th} ($j = 1, ..., N$) stream is given by $\lambda(t)^{j*} = \mu_j + \Sigma_{k=1}^{N} \Sigma_{i:\tau_i<t} \alpha_{jk} e^{(-\beta_j*(t-\tau_i))}$, where $j, k \in (1, ..., N)$. Here, α_{jk} is the excitement caused by the k^{th} stream on the j^{th}. Therefore, $\boldsymbol{\alpha}$ is an NxN matrix where the diagonals represent the within-stream excitement and the off-diagonals represent the excitement between streams.

Value

A list containing components of the fitted model, see `TMB::MakeADFun`. Includes

- `par`, a numeric vector of estimated parameter values;
- `objective`, the objective function;
- `gr`, the TMB calculated gradient function; and
- `simulate`, (where available) a simulation function.

References

Hawkes, AG. (1971) Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, **58**: 83–90.

See Also

show_hawkes

Examples

```
### ********************** ###
## A Hawkes model
### ********************** ###
data(retweets_niwa, package = "stelfi")
times <- unique(sort(as.numeric(difftime(retweets_niwa, min(retweets_niwa), units = "mins"))))
params <- c(mu = 0.05, alpha = 0.05, beta = 0.1)
fit <- fit_hawkes(times = times, parameters = params)
get_coefs(fit)
### ********************** ###
## A Hawkes model with marks (ETAS-type)
### ********************** ###
data("nz_earthquakes", package = "stelfi")
earthquakes <- nz_earthquakes[order(nz_earthquakes$origintime),]
earthquakes <- earthquakes[!duplicated(earthquakes$origintime), ]
times <- earthquakes$origintime
times <- as.numeric(difftime(times, min(times), units = "hours"))
marks <- earthquakes$magnitude
params <- c(mu = 0.05, alpha = 0.05, beta = 1)
fit <- fit_hawkes(times = times, parameters = params, marks = marks)
get_coefs(fit)


### ********************** ###
## A Hawkes process with a custom background function
### ********************** ###
if(require("hawkesbow")) {
times <- hawkesbow::hawkes(1000, fun = function(y) {1 + 0.5*sin(y)},
M = 1.5, repr = 0.5, family = "exp", rate = 2)$p
## The background function must take a single parameter and
## the time(s) at which it is evaluated
background <- function(params,times) {
```

```
A = exp(params[[1]])
B = stats::plogis(params[[2]]) * A
return(A + B  *sin(times))
}
## The background_integral function must take a
## single parameter and the time at which it is evaluated
background_integral <- function(params,x) {
        A = exp(params[[1]])
        B = stats::plogis(params[[2]]) * A
        return((A*x)-B*cos(x))
}
param = list(alpha = 0.5, beta = 1.5)
background_param = list(1,1)
fit <- fit_hawkes_cbf(times = times, parameters = param,
background = background,
background_integral = background_integral,
background_parameters = background_param)
get_coefs(fit)
}


### ********************** ###
## A multivariate Hawkes model
### ********************** ###
data(multi_hawkes, package = "stelfi")
fit <- fit_mhawkes(times = multi_hawkes$times, stream = multi_hawkes$stream,
parameters = list(mu =  c(0.2,0.2),
alpha =  matrix(c(0.5,0.1,0.1,0.5),byrow = TRUE,nrow = 2),
beta = c(0.7,0.7)))
get_coefs(fit)
```

`fit_lgcp` *Spatial or spatiotemporal log-Gaussian Cox process (LGCP)*

Description

Fit a log-Gaussian Cox process (LGCP) using Template Model Builder (TMB) and the `R_inla` namespace for the SPDE-based construction of the latent field.

Usage

```
fit_lgcp(
  locs,
  sf,
  smesh,
  tmesh,
  parameters,
  covariates,
```

```
  tmb_silent = TRUE,
  nlminb_silent = TRUE,
  ...
)
```

Arguments

`locs`	A `data.frame` of `x` and `y` locations, $2 \times n$. If a spatiotemporal model is to be fitted then there should be the third column (`t`) of the occurrence times.
`sf`	An `sf` of type `POLYGON` specifying the spatial region of the domain.
`smesh`	A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.
`tmesh`	Optional, a temporal mesh returned by `fmesher::fm_mesh_1d()`.
`parameters`	A named list of parameter starting values: • `beta`, a vector of fixed effects coefficients to be estimated, β (same length as `ncol(covariates)` + 1); • `log_tau`, the $\log(\tau)$ parameter for the GMRF; • `log_kappa`, $\log(\kappa)$ parameter for the GMRF; • `atanh_rho`, optional, $\arctan(\rho)$ AR1 temporal parameter. Default values are used if none are provided. NOTE: these may not always be appropriate.
`covariates`	Optional, a `matrix` of covariates at each `smesh` and `tmesh` node combination.
`tmb_silent`	Logical, if `TRUE` (default) then TMB inner optimisation tracing information will be printed.
`nlminb_silent`	Logical, if `TRUE` (default) then for each iteration `nlminb()` output will be printed.
`...`	optional extra arguments to pass into `stats::nlminb()`.

Details

A log-Gaussian Cox process (LGCP) where the Gaussian random field, $Z(\boldsymbol{x})$, has zero mean, variance-covariance matrix $\boldsymbol{Q}^{-1}$, and covariance function C_Z. The random intensity surface is $\Lambda(\boldsymbol{x}) = \exp(\boldsymbol{X}\beta + G(\boldsymbol{x}) + \epsilon)$, for design matrix $\boldsymbol{X}$, coefficients $\boldsymbol{\beta}$, and random error ϵ.

Shown in Lindgren et. al., (2011) the stationary solution to the SPDE (stochastic partial differential equation) $(\kappa^2 - \Delta)^{\frac{\nu + \frac{d}{2}}{2}} G(s) = W(s)$ is a random field with a Matérn covariance function, $C_Z \propto \kappa ||x - y||^{\nu} K_\nu \kappa ||x - y||$. Here ν controls the smoothness of the field and κ controls the range.

A Markovian random field is obtained when $\alpha = \nu + \frac{d}{2}$ is an integer. Following Lindgren et. al., (2011) we set $\alpha = 2$ in 2D and therefore fix $\nu = 1$. Under these conditions the solution to the SPDE is a Gaussian Markov Random Field (GMRF). This is the approximation we use.

The (approximate) spatial range $= \frac{\sqrt{8\nu}}{\kappa} = \frac{\sqrt{8}}{\kappa}$ and the standard deviation of the model, $\sigma = \frac{1}{\sqrt{4\pi\kappa^2\tau^2}}$. Under `INLA` (Lindgren and Rue, 2015) methodology the practical range is defined as the distance such that the correlation is ~ 0.139.

Value

A list containing components of the fitted model, see `TMB::MakeADFun`. Includes

- `par`, a numeric vector of estimated parameter values;
- `objective`, the objective function;
- `gr`, the TMB calculated gradient function; and
- `simulate`, a simulation function.

References

Lindgren, F., Rue, H., and Lindström, J. (2011) An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**: 423–498.

Lindgren, F. and Rue, H. (2015) Bayesian spatial modelling with R-INLA. *Journal of Statistical Software*, **63**: 1–25.

See Also

`fit_mlgcp` and `sim_lgcp`

Examples

```
### ********************** ###
## A spatial only LGCP
### ********************** ###
if(requireNamespace("fmesher")) {
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
locs <- data.frame(x = xyt$x, y = xyt$y)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd,
max.edge = 0.75, cutoff = 0.3)
fit <- fit_lgcp(locs = locs, sf = domain, smesh = smesh,
parameters = c(beta = 0, log_tau = log(1), log_kappa = log(1)))
### ********************** ###
## A spatiotemporal LGCP, AR(1)
### ********************** ###
ndays <- 2
locs <- data.frame(x = xyt$x, y = xyt$y, t = xyt$t)
w0 <- 2
tmesh <- fmesher::fm_mesh_1d(seq(0, ndays, by = w0))
fit <- fit_lgcp(locs = locs, sf = domain, smesh = smesh, tmesh = tmesh,
 parameters = c(beta = 0, log_tau = log(1), log_kappa = log(1), atanh_rho = 0.2))
}
```

fit_mlgcp *Marked spatial log-Gaussian Cox process (mLGCP)*

Description

Fit a marked LGCP using Template Model Builder (TMB) and the `R_inla` namespace for the SPDE-based construction of the latent field.

Usage

```
fit_mlgcp(
  locs,
  sf,
  marks,
  smesh,
  parameters = list(),
  methods,
  strfixed = matrix(1, nrow = nrow(locs), ncol = ncol(marks)),
  fields = rep(1, ncol(marks)),
  covariates,
  pp_covariates,
  marks_covariates,
  tmb_silent = TRUE,
  nlminb_silent = TRUE,
  ...
)
```

Arguments

`locs` A `data.frame` of `x` and `y` locations, 2xn.

`sf` An `sf` of type `POLYGON` specifying the spatial region of the domain.

`marks` A matrix of marks for each observation of the point pattern.

`smesh` A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.

`parameters` a list of named parameters: log_tau, log_kappa, betamarks, betapp, marks_coefs_pp.

`methods` An integer value:
- `0` (default), Gaussian distribution, parameter estimated is mean;
- `1`, Poisson distribution, parameter estimated is intensity;
- `2`, binomial distribution, parameter estimated is logit/probability;
- `3`, gamma distribution, the implementation in TMB is shape-scale.

`strfixed` A matrix of fixed structural parameters, defined for each event and mark. Defaults to `1`. If mark distribution
- Normal, then this is the log of standard deviation;
- Poisson, then not used;
- Binomial, then this is the number of trials;

- Gamma, then this is the log of the scale.

fields — A binary vector indicating whether there is a new random field for each mark. By default, each mark has its own random field.

covariates — Covariate(s) corresponding to each area in the spatial mesh

pp_covariates — Which columns of the covariates apply to the point process

marks_covariates — Which columns of the covariates apply to the marks. By default, all covariates apply to the marks only.

tmb_silent — Logical, if TRUE (default) then TMB inner optimisation tracing information will be printed.

nlminb_silent — Logical, if TRUE (default) then for each iteration nlminb() output will be printed.

... — optional extra arguments to pass into stats::nlminb().

Details

The random intensity surface of the point process is (as fit_lgcp) $\Lambda(\boldsymbol{x}) = \exp(\boldsymbol{X}\beta + G(\boldsymbol{x}) + \epsilon)$, for design matrix $\boldsymbol{X}$, coefficients $\boldsymbol{\beta}$, and random error ϵ.

Each mark, m_j, is jointly modelled and has their own random field $M_j(s) = f^{-1}((\boldsymbol{X}\beta)_{m_j} + G_{m_j}(\boldsymbol{x}) + \alpha_{m_j}\, G(\boldsymbol{x}) + \epsilon_{m_j})$ where $\alpha_{.}$ are coefficient(s) linking the point process and the mark(s).

$M_j(s)$ depends on the distribution of the marks. If the marks are from a Poisson distribution, it is the intensity (as with the point process). If the marks are from a Binomial distribution, it is the success probability, and the user must supply the number of trials for each event (via strfixed). If the marks are normally distributed then this models the mean, and the user must supply the standard deviation (via strfixed). The user can choose for the point processes and the marks to share a common GMRF, i.e. $G_m(s) = G_{pp}(s)$; this is controlled via the argument fields.

Value

A list containing components of the fitted model, see TMB::MakeADFun. Includes

- par, a numeric vector of estimated parameter values;
- objective, the objective function; and
- gr, the TMB calculated gradient function.

References

Lindgren, F., Rue, H., and Lindström, J. (2011) An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**: 423–498.

See Also

fit_lgcp

Examples

```
### ********************** ###
## A joint likelihood marked LGCP model
### ********************** ###
if(requireNamespace("fmesher")){
data(marked, package = "stelfi")
loc.d <- 3 * cbind(c(0, 1, 1, 0, 0), c(0, 0, 1, 1, 0))
domain <- sf::st_sf(geometry = sf::st_sfc(sf::st_polygon(list(loc.d))))
smesh <- fmesher::fm_mesh_2d(loc.domain = loc.d, offset = c(0.3, 1),
max.edge = c(0.3, 0.7), cutoff = 0.05)
locs <- cbind(x = marked$x, y = marked$y)
marks <- cbind(m1 = marked$m1) ## Gaussian mark
parameters <- list(betamarks = matrix(0, nrow = 1, ncol = ncol(marks)),
log_tau = rep(log(1), 2), log_kappa = rep(log(1), 2),
marks_coefs_pp = rep(0, ncol(marks)), betapp = 0)
fit <- fit_mlgcp(locs = locs, marks = marks,
sf = domain, smesh = smesh,
parameters = parameters, methods = 0,fields = 1)
}
```

`fit_stelfi` *Modelling spatiotemporal self-excitement*

Description

Fits spatiotemporal Hawkes models. The self-excitement is Gaussian in space and exponentially decaying in time.

Usage

```
fit_stelfi(
  times,
  locs,
  sf,
  smesh,
  parameters,
  covariates,
  GMRF = FALSE,
  time_independent = TRUE,
  tmb_silent = TRUE,
  nlminb_silent = TRUE,
  ...
)
```

Arguments

- `times` — A vector of numeric observed time points.
- `locs` — A `data.frame` of `x` and `y` locations, 2xn.
- `sf` — An `sf` of type `POLYGON` specifying the spatial region of the domain.
- `smesh` — A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.
- `parameters` — A list of named parameters:
  - `coefs`, logged base rate of the Hawkes process and coefficients of covariates
  - `alpha`, intensity jump after an event occurrence
  - `beta`, rate of exponential decay of intensity after event occurrence
  - `tau`, τ parameter for the GMRF (supplied only if `GMRF = TRUE`)
  - `kappa`, κ parameter for the GMRF (supplied only if `GMRF = TRUE`)
  - `xsigma`, standard deviation on x-axis of self-exciting kernel (if `time_independent = FALSE`, it is the s.d. after 1 time unit)
  - `ysigma`, standard deviation on y-axis of self-exciting kernel (if `time_independent = FALSE`, it is the s.d. after 1 time unit)
  - `rho`, correlation between x and y for the self-exciting kernel (the off-diagonal elements in the kernel's covariate matrix are `xsigma * ysigma * rho`)
- `covariates` — Optional, a `matrix` of covariates at each `smesh` node.
- `GMRF` — Logical, default `FALSE`. If `TRUE`, a Gaussian Markov Random Field is included as a latent spatial effect.
- `time_independent` — Logical, default `TRUE`. If `FALSE`, Gaussian kernels have a covariate matrix that is proportional to time since the event. Warning, this is very memory intensive.
- `tmb_silent` — Logical, if `TRUE` (default) then TMB inner optimisation tracing information will be printed.
- `nlminb_silent` — Logical, if `TRUE` (default) then for each iteration `nlminb()` output will be printed.
- `...` — Additional arguments to pass to `optim()`

Details

Temporal self-excitement follows an exponential decay function. The self-excitement over space follows a Gaussian distribution centered at the triggering event. There are two formulations of this model. The default is that the Gaussian function has a fixed spatial covariance matrix, independent of time. Alternatively, covariance can be directly proportional to time, meaning that the self-excitement radiates out from the center over time. This can be appropriate when the mechanism causing self-excitement travels at a finite speed, but is very memory-intensive. The spatiotemporal intensity function used by `stelfi` is $\lambda(s,t) = \mu + \alpha \Sigma_{i:\tau_i<t}(\exp(-\beta * (t-\tau_i))G_i(s-x_i, t-\tau_i))$ where

- μ is the background rate,
- β is the rate of temporal decay,
- α is the increase in intensity after an event,

- τ_i are the event times,
- x_i are the event locations (in 2D Euclidean space), and
- $G_i(s - x_i, t - \tau_i)$ is the spatial self-excitement kernel.

$G_i(.,.)$ can take two forms:

- For time-independent spatial excitement (`time_independent = TRUE`), $G_i(s - x_i, t - \tau_i) = f(s - x_i)$ where f is the density function of $\mathrm{N}(0, \Sigma)$.
- For time-dependent spatial excitement (`time_independent = FALSE`), $G_i(s - x_i, t - \tau_i) = f(s - x_i)$ where f is the density function of $\mathrm{N}(0, (t - \tau_i)\Sigma)$.

Value

A list containing components of the fitted model, see `TMB::MakeADFun`. Includes

- `par`, a numeric vector of estimated parameter values;
- `objective`, the objective function; and
- `gr`, the TMB calculated gradient function.

See Also

`fit_hawkes` and `fit_lgcp`

Examples

```
## No GMRF
if(requireNamespace("fmesher")){
data(xyt, package = "stelfi")
N <- 50
locs <- data.frame(x = xyt$x[1:N], y = xyt$y[1:N])
times <- xyt$t[1:N]
domain <- sf::st_as_sf(xyt$window)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd, max.edge = 0.75, cutoff = 0.3)
param <- list( mu = 3, alpha = 1, beta = 3, xsigma = 0.2, ysigma = 0.2, rho = 0.8)
fit <- fit_stelfi(times = times, locs = locs, sf = domain, smesh = smesh, parameters = param)
get_coefs(fit)
## GMRF
param <- list( mu = 5, alpha = 1, beta = 3, kappa = 0.9, tau = 1, xsigma = 0.2,
ysigma = 0.2, rho = 0.8)
fit <- fit_stelfi(times = times, locs = locs, sf = domain, smesh = smesh,
parameters = param, GMRF = TRUE)
get_coefs(fit)
}
```

get_coefs *Extract reported parameter estimates*

Description

Return parameter estimates from a fitted model.

Usage

```
get_coefs(obj)
```

Arguments

obj	A fitted model as returned by one of fit_hawkes, fit_hawkes_cbf, fit_lgcp, fit_mlgcp, or fit_stelfi.

Value

A matrix of estimated parameters and standard errors returned by TMB::sdreport ("report").

See Also

fit_hawkes, fit_hawkes_cbf, fit_lgcp, fit_mlgcp, and fit_stelfi

Examples

```
## Hawkes
data(retweets_niwa, package = "stelfi")
times <- unique(sort(as.numeric(difftime(retweets_niwa, min(retweets_niwa),units = "mins"))))
params <- c(mu = 9, alpha = 3, beta = 10)
fit <- fit_hawkes(times = times, parameters = params)
get_coefs(fit)
## LGCP
if(requireNamespace("fmesher")) {
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
locs <- data.frame(x = xyt$x, y = xyt$y)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd, max.edge = 0.75, cutoff = 0.3)
fit <- fit_lgcp(locs = locs, sf = domain, smesh = smesh,
parameters = c(beta = 0, log_tau = log(1), log_kappa = log(1)))
get_coefs(fit)
}
```

get_fields *Estimated random field(s)*

Description

Extract the estimated mean, or standard deviation, of the values of the Gaussian Markov random field for a fitted log-Gaussian Cox process model at each node of `smesh`.

Usage

```
get_fields(obj, smesh, tmesh, plot = FALSE, sd = FALSE)
```

Arguments

obj	A fitted model object returned by `fit_lgcp`.
smesh	A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.
tmesh	Optional, a temporal mesh returned by `fmesher::fm_mesh_1d()`.
plot	Logical, if `TRUE` then the returned values are plotted. Default `FALSE`.
sd	Logical, if `TRUE` then standard errors returned. Default `FALSE`.

Value

A `numeric` vector or a `list` of returned values at each `smesh` node.

See Also

`fit_lgcp` and `fit_mlgcp`

Examples

```
if(requireNamespace("fmesher")) {
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
locs <- data.frame(x = xyt$x, y = xyt$y)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd, max.edge = 0.75, cutoff = 0.3)
fit <- fit_lgcp(locs = locs, sf = domain, smesh = smesh,
parameters = c(beta = 0, log_tau = log(1), log_kappa = log(1)))
get_fields(fit, smesh, plot = TRUE)
}
```

get_weights *Mesh weights*

Description

Calculate the areas (weights) around the mesh nodes that are within the specified spatial polygon `sf` of the domain.

Usage

```
get_weights(mesh, sf, plot = FALSE)
```

Arguments

`mesh`	A spatial mesh of class `fmesher::fm_mesh_2d()`
`sf`	An `sf` of type `POLYGON` specifying the region of the domain.
`plot`	Logical, whether to plot the calculated `mesh` weights. Default, `FALSE`.

Value

Either a simple features, `sf`, object or values returned by `geom_sf`.

See Also

https://becarioprecario.bitbucket.io/spde-gitbook/.

Examples

```
data(horse_mesh, package = "stelfi")
data(horse_sf, package = "stelfi")
get_weights(horse_mesh, horse_sf, plot = TRUE)
```

horse_mesh *Example Delaunay triangulation*

Description

Example Delaunay triangulation

Format

A `fmesher::fm_mesh_2d()` based on the outline of a horse

horse_sf — *Example* sf POLYGON

Description

Example sf POLYGON

Format

A sf POLYGON of a horse outline

iraq_terrorism — *Terrorism in Iraq, 2013 - 2017*

Description

A dataset containing information of terrorism activity carried out by the Islamic State of Iraq and the Levant (ISIL) in Iraq, 2013 - 2017.

Format

A simple features dataframe, of type POINT, with 4208 observations and 16 fields:

iyear numeric year 2013–2017

imonth numeric month index 1–12

iday numeric day 1–31 (zeros are a non-entry)

country country (IRAQ)

latitude latitude location

longitude longitude location

utm_x x-coord location UTM

utm_y y-coord location UTM

success logical, was fatal? TRUE = fatal

nkill number of fatalities per attack

specificity spatial accuracy of event: 1 = most accurate, 5 = worst

gname character name of attack perpetrators (ISIL)

x_coord x coordinate from location projected onto a sphere

y_coord y coordinate from location projected onto a sphere

z_coord z coordinate from location projected onto a sphere

popdensity scaled: number of people per kilometer squared

luminosity scaled: luminosity

tt scaled: time to nearest city in minutes

Source

https://www.start.umd.edu/gtd/

---

marked *Example marked point pattern data set*

Description

Example marked point pattern data set

Format

A data frame with 159 rows and 5 variables:

x x coordinate

y y coordinate

m1 mark, Gaussian distributed

m2 mark, Bernoulli distributed

m3 mark, Gamma distributed

---

meshmetrics *Calculate a number of different geometric attributes of a Delaunay triangulation*

Description

Calculates a number of geometric attributes for a given Delaunay triangulation based on the circumscribed and inscribed circle of each triangle.

Usage

```
meshmetrics(mesh)
```

Arguments

mesh	A `fmesher::fm_mesh_2d` object.

Details

A triangle's circumcircle (circumscribed circle) is the unique circle that passes through each of its three vertices. A triangle's incircle (inscribed circle) is the largest circle that can be contained within it (i.e., touches it's three edges).

Value

An object of class `sf` with the following data for each triangle in the triangulation

- `V1`, `V2`, and `V3` corresponding vertices of `mesh` matches `mesh$graph$tv`;
- `ID`, numeric triangle id;
- `angleA`, `angleB`, and `angleC`, the interior angles;
- circumcircle radius, circumradius, `circumcircle_R` (R);
- incircle radius `incircle_r` (r);
- centroid locations of the circumcircle, circumcenter, (`c_Ox, c_Oy`);
- centroid locations of the incircle, incenter, (`i_Ox, i_Oy`);
- the radius-edge ratio `radius_edge` $\frac{R}{l_{min}}$, where l_{min} is the minimum edge length;
- the radius ratio `radius_ratio` $\frac{r}{R}$;
- `area`, area (A);
- `quality` a measure of "quality" defined as $\frac{4\sqrt{3}|A|}{\Sigma_{i=1}^{3}L_i^2}$, where L_i is the length of edge i.

Examples

```
data(horse_mesh, package = "stelfi")
metrics <- meshmetrics(horse_mesh)
if(require("ggplot2")) {
ggplot(metrics) + geom_sf(aes(fill = radius_ratio))
}
```

`mesh_2_sf` *Transform a* `fmesher::fm_mesh_2d` *into a* `sf` *object*

Description

Transform a `fmesher::fm_mesh_2d` into a `sf` object

Usage

```
mesh_2_sf(mesh)
```

Arguments

`mesh`	A `fmesher::fm_mesh_2d` object.

Value

A simple features, `sf`, object.

Source

Modified from sp based function suggested by Finn in the R-inla discussion Google Group https://groups.google.com/g/r-inla-discussion-group/c/z1n1exlZrKM.

See Also

meshmetrics

Examples

```
data(horse_mesh, package = "stelfi")
sf <- mesh_2_sf(horse_mesh)
if(require("ggplot2")) {
ggplot(sf) + geom_sf()
}
```

---

multi_hawkes *Example multivariate Hawkes dataset*

Description

Two-stream multivariate Hawkes data with $mu_1 = \mu_2 = 0.2$, $beta_1 = \beta_2 = 0.7$, and $\boldsymbol{\alpha}$ = `matrix(c(0.5,0.1,0.1,0.5),byrow = TRUE,nrow = 2)`.

Format

A data frame with 213 observations and 2 variables:

times ordered time stamp of observation

stream character giving stream ID (i.e., Stream 1 or Stream 2) of observation

---

nz_earthquakes *Earthquakes in Canterbury, NZ, 2010 - 2014*

Description

Earthquake data from Canterbury, New Zealand 16-Jan-2010–24-Dec-2014.

Format

A simple features dataframe, of type `POINT`, with 3824 observations and 3 fields:

origintime The UTC time of the event's occurrence

magnitude The magnitude of the earthquake

depth The focal depth of the event (km)

Source

https://quakesearch.geonet.org.nz/

`nz_murders` *Murders of NZ, 2004 - 2019*

Description

A dataset of recorded murder cases in New Zealand 2004 - 2019.

Format

A simple features dataframe, of type `POINT`, with 967 observations and 11 fields:

sex Biological sex of victim

age Age of victim (years)

date Month and day of murder

year Year

cause Cause of death

killer Killer

name Name of victim

full_date Date object of observation on single days

month Month name of observation

cause_cat Cause of death as category

region NZ region

Source

Data scraped and cleaned by Charlie Timmings, honours student at the University of Auckland from the website https://interactives.stuff.co.nz/2019/the-homicide-report/

`retweets_niwa` *Retweets of NIWA's viral leopard seal Tweet*

Description

A dataset of retweet times of NIWA's viral leopard seal tweet on the 5th Feb 2019 (https://twitter.com/niwa_nz/status/1092610541401587712).

Format

A vector of length 4890 specifying the date and time of retweet (UTC)

Source

https://twitter.com/niwa_nz/status/1092610541401587712 sasquatch *Sasquatch (bigfoot) sightings in the USA, 2000 - 2005*

Description

Subset of data sourced from the Bigfoot Field Researchers Organization (BFRO) (https://data.world/timothyrenner/bfro-sightings-data).

Format

A simple features dataframe, of type `POINT`, with 972 observations and 27 fields:

observed Text observation summary

location_details Text location summary

county County

state State

season Season

title Report title

date Date

number Report number

classification Report classification

geohash Geohash code

temperature_high Reported weather measure

temperature_mid Reported weather measure

temperature_low Reported weather measure

dew_point Reported weather measure

humidity Reported weather measure

cloud_cover Reported weather measure

moon_phase Reported measure

precip_intensity Reported weather measure

precip_probability Reported weather measure

precip_type Reported weather measure

pressure Reported weather measure

summary Text weather summary

uv_index Reported weather measure

visibility Reported weather measure

wind_bearing Reported weather measure

wind_speed Reported weather measure

year Year

Source

https://data.world/timothyrenner/bfro-sightings-data `show_field` *Plot the estimated random field(s) of a fitted LGCP*

Description

Plots the values of `x` at each node of `smesh`, with optional control over resolutions using `dims`.

Usage

```
show_field(x, smesh, sf, dims = c(500, 500), clip = FALSE)
```

Arguments

`x`	A vector of values, one value per each `smesh` node.
`smesh`	A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.
`sf`	Optional, `sf` of type `POLYGON` specifying the region of the domain.
`dims`	A numeric vector of length 2 specifying the spatial pixel resolution. Default `c(500,500)`.
`clip`	Logical, if `TRUE` then plotted values are ‘clipped‘ to the domain supplied as `sf`.

Value

A gg class object, values returned by `geom_tile` and `geom_sf`.

See Also

`show_lambda` and `get_fields`

Examples

```
if(requireNamespace("fmesher")){
if(require("sf")){
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd, max.edge = 0.75, cutoff = 0.3)
parameters <- c(beta = 1, log_tau = log(1), log_kappa = log(1))
simdata <- sim_lgcp(parameters = parameters, sf = domain, smesh = smesh)
show_field(c(simdata$x), smesh = smesh, sf = domain)
show_field(c(simdata$x), smesh = smesh, sf = domain, clip = TRUE)
}
}
```

show_hawkes *Plot Hawkes intensity*

Description

Plots a Hawkes intensity function, options to extend to non-homogeneous background intensity.

Plots a number of goodness-of-fit plots for a fitted Hawkes process. Includes 1) a comparison of the compensator and observed events, 2) a histogram of transformed interarrival times, 3) a Q-Q plot of transformed interarrival times, and 4) the CDF of consecutive interarrival times, In addition, results of a Kolmogorov-Smirnov and Ljung-Box hypothesis test for the interarrival times are printed.

Usage

```
show_hawkes(obj)

show_hawkes_GOF(
  obj,
  background_integral = NULL,
  plot = TRUE,
  return_values = FALSE
)
```

Arguments

obj	Either object returned by fit_hawkes/fit_hawkes_cbf or a named list with elements times and params. If the latter, then times should be a numeric vector of observed time points, and params must contain, alpha (intensity jump after an event occurrence) and beta (exponential intensity decay). In addition, should contain either mu (base rate of the Hawkes process) or background_parameters (parameter(s) for the assumed non-homogeneous background function; could be a list of multiple values). May also contain marks (a vector of numerical marks).
background_integral	Function, as defined in fit_hawkes_cbf.
plot	Logical, whether to plot goodness-of-fit plots. Default TRUE.
return_values	Logical, whether to return GOF values. Default FALSE.

Value

show_hawkes returns a gtable object with geom_line and geom_histogram values.

show_hawkes_GOF returns no value unless return_values = TRUE, in this case a list of interarrival times is returned.

Examples

```
data(retweets_niwa, package = "stelfi")
times <- unique(sort(as.numeric(difftime(retweets_niwa, min(retweets_niwa),units = "mins"))))
params <- c(mu = 9, alpha = 3, beta = 10)
show_hawkes(list(times = times, params = params))
fit <- fit_hawkes(times = times, parameters = params)
show_hawkes(fit)
data(retweets_niwa, package = "stelfi")
times <- unique(sort(as.numeric(difftime(retweets_niwa, min(retweets_niwa),units = "mins"))))
params <- c(mu = 9, alpha = 3, beta = 10)
show_hawkes_GOF(list(times = times, params = params))
fit <- fit_hawkes(times = times, parameters = params)
show_hawkes_GOF(fit)
```

---

show_lambda *Plot the estimated intensity from a fitted LGCP model*

Description

Plots the estimated spatial intensity from a fitted log-Gaussian Cox process model. If `obj` is a spatiotemporal model then `timestamp` provides control over which temporal index to plot the estimated spatial intensity.

Usage

```
show_lambda(
  obj,
  smesh,
  sf,
  tmesh,
  covariates,
  clip = FALSE,
  dims = c(500, 500),
  timestamp = 1
)
```

Arguments

`obj`	A fitted LGCP model object for, for example, `fit_lgcp()`.
`smesh`	A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.
`sf`	An `sf` of type `POLYGON` specifying the spatial region of the domain.
`tmesh`	Optional, a temporal mesh returned by `fmesher::fm_mesh_1d()`.
`covariates`	Optional, a `matrix` of covariates at each `smesh` and `tmesh` node combination.
`clip`	Logical, if `TRUE` then plotted values are ‘clipped‘ to the domain supplied as `sf`.
`dims`	A numeric vector of length 2 specifying the spatial pixel resolution. Default `c(500,500)`.
`timestamp`	The index of time stamp to plot. Default `1`.

Value

A gg class object, values returned by geom_tile and geom_sf.

See Also

fit_lgcp, show_field, and get_fields

Examples

```
if(requireNamespace("fmesher")) {
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
locs <- data.frame(x = xyt$x, y = xyt$y)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd, max.edge = 0.75, cutoff = 0.3)
fit <- fit_lgcp(locs = locs, sf = domain, smesh = smesh,
parameters = c(beta = 0, log_tau = log(1), log_kappa = log(1)))
show_lambda(fit, smesh = smesh, sf = domain)
}
```

---

sim_hawkes *Simulate a self-exciting Hawkes process*

Description

Simulates a self-exciting Hawkes process.

Usage

```
sim_hawkes(mu, alpha, beta, n = 100, plot = FALSE, seed = 123, method = "1")
```

Arguments

mu	A numeric specifying base rate of the Hawkes process.
alpha	A numeric specifying intensity jump after an event occurrence.
beta	A numeric specifying exponential intensity decay
n	A numeric depending on method: if method = "1" specifies end of the time line within which to simulate the process, if method = "2" specifies the number of observations to simulate. Default, 100.
plot	Logical, if TRUE data plotted along with the intensity. Default, FALSE.
seed	The seed. Default, 123
method	A character "1" or "2" specifying the method (see details) to simulate Hawkes process. Default,"1".

Details

Option of two methods to simulate a Hawkes process: if `method = "1"` then a univariate Hawkes process as `hawkes::simulateHawkes()` is simulated, if `method = "2"` then an accept/reject framework is used.

Value

A `numeric` vector of simulated event times.

See Also

`fit_hawkes`

Examples

```
sim_hawkes(10.2, 3.1, 8.9)
sim_hawkes(10.2, 3.1, 8.9, method = "2")
```

---

sim_lgcp — *Simulate a log-Gaussian Cox process (LGCP)*

Description

Simulate a realisation of a log-Gaussian Cox process (LGCP) using the `TMB` `C++` template. If `rho` is supplied in `parameters` as well as `tmesh` then spatiotemporal (AR(1)) data will be simulated.

Usage

```
sim_lgcp(parameters, sf, smesh, tmesh, covariates, all = FALSE)
```

Arguments

`parameters` A named list of parameter starting values:

- `beta`, a vector of fixed effects coefficients to be estimated, β (same length as `ncol(covariates)` + 1);
- `log_tau`, the $\log(\tau)$ parameter for the GMRF;
- `log_kappa`, $\log(\kappa)$ parameter for the GMRF;
- `atanh_rho`, optional, $\arctan(\rho)$ AR1 temporal parameter.

Default values are used if none are provided. NOTE: these may not always be appropriate.

`sf` An `sf` of type `POLYGON` specifying the spatial region of the domain.

`smesh` A Delaunay triangulation of the spatial domain returned by `fmesher::fm_mesh_2d()`.

`tmesh` Optional, a temporal mesh returned by `fmesher::fm_mesh_1d()`.

`covariates` Optional, a `matrix` of covariates at each `smesh` and `tmesh` node combination.

`all` Logical, if `TRUE` then all model components are returned.

Value

A named list. If all = FALSE then only the simulated values of the GMRF at each mesh node are returned, x, alongside the number of events, y, simulated at each node.

See Also

fit_lgcp

Examples

```
if(requireNamespace("fmesher")){
if(require("sf")) {
data(xyt, package = "stelfi")
domain <- sf::st_as_sf(xyt$window)
bnd <- fmesher::fm_as_segm(as.matrix(sf::st_coordinates(domain)[, 1:2]))
smesh <- fmesher::fm_mesh_2d(boundary = bnd,
max.edge = 0.75, cutoff = 0.3)
parameters <- c(beta = 1, log_tau = log(1), log_kappa = log(1))
sim <- sim_lgcp(parameters = parameters, sf = domain, smesh = smesh)
## spatiotemporal
ndays <- 2
w0 <- 2
tmesh <- fmesher::fm_mesh_1d(seq(0, ndays, by = w0))
parameters <- c(beta = 1, log_tau = log(1), log_kappa = log(1), atanh_rho = 0.2)
sim <- sim_lgcp(parameters = parameters, sf = domain, smesh = smesh, tmesh = tmesh)
}
}
```

---

stelfi — *A package to fit Hawkes and Log-Gaussian Cox Point Process models using Template Model Builder*

Description

Fit Hawkes and log-Gaussian Cox process models with extensions. Introduced in Hawkes (1971) a Hawkes process is a self-exciting temporal point process where the occurrence of an event immediately increases the chance of another. We extend this to consider self-inhibiting process and a non-homogeneous background rate. A log-Gaussian Cox process is a Poisson point process where the log-intensity is given by a Gaussian random field. We extend this to a joint likelihood formulation fitting a marked log-Gaussian Cox model. In addition, the package offers functionality to fit self-exciting spatiotemporal point processes. Models are fitted via maximum likelihood using 'TMB' (Template Model Builder) (Kristensen, Nielsen, Berg, Skaug, and Bell, 2016). Where included 1) random fields are assumed to be Gaussian and are integrated over using the Laplace approximation and 2) a stochastic partial differential equation model, introduced by Lindgren, Rue, and Lindström. (2011), is defined for the field(s).

Model fitting

- The functions fit_hawkes and fit_hawkes_cbf fit self-exciting Hawkes (Hawkes AG., 1971) processes to temporal point pattern data.
- The function fit_lgcp fit a log-Gaussian Cox process to either spatial or spatiotemporal point pattern data. If a spatiotemporal model is fitted a AR1 process is assumed for the temporal progression.
- The function fit_mlgcp fits a joint likelihood model between the point locations and the mark(s).
- The function fit_stelfi fits self-exciting spatiotemporal Hawkes models to point pattern data. The self-excitement is Gaussian in space and exponentially decaying over time. In addition, a GMRF can be included to account for latent spatial dependency.

References

Hawkes, AG. (1971) Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, **58**: 83–90.

Lindgren, F., Rue, H., and Lindström, J. (2011) An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**: 423–498.

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., and Bell B. M. (2016). TMB: Automatic Differentiation and Laplace Approximation. *Journal of Statistical Software*, **70**: 1–21.

---

uk_serial *Serial killers of the UK, 1828 - 2015*

Description

A dataset containing the names and number of recorded murders committed by some of the infamous UK serial killers from 1828 to 2015.

Format

A dataframe with 62 rows and 8 variables:

number_of_kills approx number of murders committed

years The years of operation

name Name of convicted serial killer

aka Some serial killers were given nicknames

year_start Year the murders began

tear_end Year the murders ended

date_of_first_kill Date, if known, first murder was committed

population_million Est popn in millions at time of first murder

Source

https://www.murderuk.com/

xyt *Self-exciting point pattern*

Description

Simulated self-exciting spatiotemporal point pattern of class `stppp`

Format

A `stppp` object with 653 observations

window Domain of the point pattern of class `owin`

n Number of observations, 653

x x coordinate

y y coordinate

markformat `none`

t Timestamp of points

tlim Time frame `0 2`

Index

fit_hawkes, 3, *13, 14*, *24*, *27*, *29*
fit_hawkes_cbf, *14*, *24*, *29*
fit_hawkes_cbf *(*fit_hawkes*)*, 3
fit_lgcp, 6, *10*, *13–15*, *26*, *28, 29*
fit_mhawkes *(*fit_hawkes*)*, 3
fit_mlgcp, *8*, 9, *14, 15*, *29*
fit_stelfi, 11, *14*, *29*

get_coefs, 14
get_fields, 15, *23*, *26*
get_weights, 16

horse_mesh, 16
horse_sf, 17

iraq_terrorism, 17

marked, 18
mesh_2_sf, 19
meshmetrics, 18, *20*
multi_hawkes, 20

nz_earthquakes, 20
nz_murders, 21

retweets_niwa, 21

sasquatch, 22
show_field, 23, *26*
show_hawkes, *5*, *24*, 24
show_hawkes_GOF, *24*
show_hawkes_GOF *(*show_hawkes*)*, 24
show_lambda, *23*, 25
sim_hawkes, 26
sim_lgcp, *8*, 27
stelfi, *12*, 28
stelfi-package *(*stelfi*)*, 28

uk_serial, 29

xyt, 30