Package 'binspp'

May 7, 2026

Type Package

Title Bayesian Inference for Neyman-Scott Point Processes

Version 0.2.3

Description The Bayesian MCMC estimation of parameters for Thomas-type cluster point process with various inhomogeneities. It allows for inhomogeneity in (i) distribution of parent points, (ii) mean number of points in a cluster, (iii) cluster spread. The package also allows for the Bayesian MCMC algorithm for the homogeneous generalized Thomas process. The cluster size is allowed to have a variance that is greater or less than the expected value (cluster sizes are over or under dispersed). Details are described in Dvořák, Remeš, Beránek & Mrkvička (2022) <arXiv: 10.48550/arXiv.2205.07946>.

License GPL-3

URL https://github.com/tomasmrkvicka/binspp

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports Rcpp, VGAM, cluster, mvtnorm, spatstat, spatstat.model, spatstat.geom, spatstat.random, fields, stats

LinkingTo Rcpp, RcppArmadillo, RcppEigen

RoxygenNote 7.3.3

NeedsCompilation yes

Author Mrkvicka Tomas [aut], Dvorak Jiri [aut], Beranek Ladislav [aut], Remes Radim [aut, cre], Park Jaewoo [ctb], Lee Sujeong [ctb]

Maintainer Remes Radim <inrem@jcu.cz>

Repository CRAN

Date/Publication 2025-12-03 23:40:07 UTC

Contents

`AuxVarGen` *Generate auxiliary variable for given proposed parameters.*

Description

Generate auxiliary variable for given proposed parameters.

Usage

```
AuxVarGen(kappa, theta, likelihoodprev, rho0sum, CC, AreaMRW, W_dil, niter)
```

Arguments

kappa	parameter to generate auxiliary variable. κ represents the -.
theta	parameter vector $\Theta = (\theta_1, \theta_2).\theta_1 represents - .\theta_2 represents - \, .$
likelihoodprev	previous likelihood value.
rho0sum	Initial sum of interaction strengths.
CC	cluster centers.
AreaMRW	area of dilated window.
W_dil	observation window dilated by the assumed maximal cluster radius.
niter	number of iterations of MCMC.

Value

The output is a list of CC, likelihoodprev, rho0sum.

---

binspp *Bayesian inference for Neyman-Scott point processes*

Description

The Bayesian MCMC estimation of parameters for Thomas-type cluster point process with various inhomogeneities. It allows for inhomogeneity in (i) distribution of parent points, (ii) mean number of points in a cluster, (iii) cluster spread. The package also allows for the Bayesian MCMC algorithm for the homogeneous generalized Thomas process. The cluster size is allowed to have a variance that is greater or less than the expected value (cluster sizes are over or under dispersed). Details are described in Dvořák, Remeš, Beránek & Mrkvička (2022) (doi:10.48550/arXiv.2205.07946).

Note

License: GPL-3

Author(s)

Tomas Mrkvicka <mrkvicka.toma@gmail.com> (author), Jiri Dvorak <dvorak@karlin.mff.cuni.cz> (author), Ladislav Beranek <beranek@jcu.cz> (author), Radim Remes <inrem@jcu.cz> (author, creator), Jaewoo Park <jwpark88@yonsei.ac.kr> (contributor), Sujeong Lee <dltnwjd2304@gmail.com> (contributor)

References

Anderson, C. Mrkvička T. (2020). Inference for cluster point processes with over- or under-dispersed cluster sizes, *Statistics and computing* **30**, 1573–1590, doi:10.1007/s11222020099608.

Kopecký J., Mrkvička T. (2016). On the Bayesian estimation for the stationary Neyman-Scott point processes, Applications of Mathematics **61/4**, 503-514. Available from: doi:10.1007/s10492016-01448.

Dvořák, J., Remeš, R., Beránek, L., Mrkvička, T. (2022). binspp: An R Package for Bayesian Inference for Neyman-Scott Point Processes with Complex Inhomogeneity Structure. *arXiv*. doi:10.48550/ARXIV.2205.07946.

`coeff` *Calculate parameters for Birth and Death Interaction likelihood functions.*

Description

Calculates r1 and r2, which serve as distance thresholds for Birth and Death Interaction likelihood functions.

Usage

```
coeff(theta)
```

Arguments

`theta`	A vector of theta1 and theta2.

Value

A vector of r1 and r2.

`cov_refor` *Distance to the reforestration polygon*

Description

Covariate for data trees_N4.

Usage

```
cov_refor
```

Format

An object of class `im` with 208 rows and 374 columns.

cov_reserv *Distance to the reservoir*

Description

Covariate for data trees_N4.

Usage

```
cov_reserv
```

Format

An object of class `im` with 208 rows and 374 columns.

cov_slope *Slope of the area*

Description

Covariate for data trees_N4.

Usage

```
cov_slope
```

Format

An object of class `im` with 208 rows and 374 columns.

cov_tdensity *Trees density*

Description

Covariate for data trees_N4.

Usage

```
cov_tdensity
```

Format

An object of class `im` with 208 rows and 374 columns.

cov_tmi *Topographic moisture index*

Description

Covariate for data trees_N4.

Usage

```
cov_tmi
```

Format

An object of class `im` with 208 rows and 374 columns.

estgtp *Auxiliary function which calculates sum values Bayesian MCMC estimation of parameters of generalized Thomas process*

Description

Bayesian MCMC estimation of parameters of generalized Thomas process. The cluster size is allowed to have a variance that is greater or less than the expected value (cluster sizes are over or under dispersed).

Usage

```
estgtp(
  X,
  kappa0 = exp(a_kappa + ((b_kappa^2)/2)),
  omega0 = exp(a_omega + ((b_omega^2)/2)),
  lambda0 = (l_lambda + u_lambda)/2,
  theta0 = exp(a_theta + ((b_theta^2)/2)),
  skappa,
  somega,
  dlambda,
  stheta,
  smove,
  a_kappa,
  b_kappa,
  a_omega,
  b_omega,
  l_lambda,
  u_lambda,
  a_theta,
```

```
  b_theta,
  iter = 5e+05,
  plot.step = 1000,
  save.step = 1000,
  filename
)
```

Arguments

X	A point pattern dataset (object of class *ppp*) to which the model should be fitted.
kappa0	Initial value for *kappa*, by default it will be set as expectation of prior for *kappa*.
omega0	Initial value for *omega*, by default it will be set as expectation of prior for *omega*.
lambda0	Initial value for *lambda*, by default it will be set as expectation of prior for *lambda*.
theta0	Initial value for *theta*, by default it will be set as expectation of prior for *theta*.
skappa	variability of proposal for *kappa*: second parameter of log-normal distribution
somega	variability of proposal for *omega*: second parameter of log-normal distribution
dlambda	variability of proposal for *lambda*: half of range of uniform distribution
stheta	variability of proposal for *theta*: second parameter of log-normal distribution
smove	variability of proposal for moving center point: SD of normal distribution
a_kappa	First parameter of prior distribution for *kappa*, which is log-normal distribution.
b_kappa	Second parameter of prior distribution for *kappa*, which is log-normal distribution.
a_omega	First parameter of prior distribution for *omega*, which is log-normal distribution.
b_omega	Second parameter of prior distribution for omega, which is log-normal distribution.
l_lambda	First parameter of prior distribution for *lambda*, which is uniform distribution.
u_lambda	Second parameter of prior distribution for *lambda*, which is uniform distribution.
a_theta	First parameter of prior distribution for *theta*, which is log-normal distribution.
b_theta	Second parameter of prior distribution for *theta*, which is log-normal distribution.
iter	Number of iterations of MCMC.
plot.step	Step for the graph plotting. If the value is greater than *iter* parameter value, no plots will be visible.
save.step	Step for the parameters saving. The file must be specified or has to be set to larger than *iter*.
filename	The name of the output RDS file

Value

The output is an estimated MCMC chain of parameters, centers and connections.

Examples

```
library(spatstat)
kappa = 10
omega = .1
lambda= .5
theta = 10

X = rgtp(kappa, omega, lambda, theta, win = owin(c(0, 1), c(0, 1)))
plot(X$X)
plot(X$C)

a_kappa = 4
b_kappa = 1
x <- seq(0, 100, length = 100)
hx <- dlnorm(x, a_kappa, b_kappa)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

a_omega = -3
b_omega = 1
x <- seq(0, 1, length = 100)
hx <- dlnorm(x, a_omega, b_omega)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

l_lambda = -1
u_lambda = 0.99
x <- seq(-1, 1, length = 100)

hx <- dunif(x, l_lambda, u_lambda)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

a_theta = 4
b_theta = 1
x <- seq(0, 100, length = 100)
hx <- dlnorm(x, a_theta, b_theta)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

est = estgtp(X$X,
          skappa = exp(a_kappa + ((b_kappa ^ 2) / 2)) / 100,
          somega = exp(a_omega + ((b_omega ^ 2) / 2)) / 100,
          dlambda = 0.01,
          stheta = exp(a_theta + ((b_theta ^ 2) / 2)) / 100, smove = 0.1,
          a_kappa = a_kappa, b_kappa = b_kappa,
          a_omega = a_omega, b_omega = b_omega,
          l_lambda = l_lambda, u_lambda = u_lambda,
          a_theta = a_theta, b_theta = b_theta,
          iter = 50, plot.step = 50, save.step = 1e9,
          filename = "")
```

estgtpr *Results for Bayesian MCMC estimation of parameters of generalized Thomas process*

Description

Calculates median values for *kappa*, *omega*, *lambda*, *theta*; calculates 2.5 and 97.5 quantile and draws trace plots.

Usage

```
estgtpr(est, discard = 100, step = 10)
```

Arguments

est	Output from estgtp() function.
discard	Number of iterations to be discarded as burn in for the estimation.
step	Every *step* iteration is taken in the parameter estimation.

Value

Median and quantile values and plots (*kappa*, *omega*, *lambda*, *theta*).

Examples

```
library(spatstat)
kappa = 10
omega = .1
lambda= .5
theta = 10

X = rgtp(kappa, omega, lambda, theta, win = owin(c(0, 1), c(0, 1)))
plot(X$X)
plot(X$C)

a_kappa = 4
b_kappa = 1
x <- seq(0, 100, length = 100)
hx <- dlnorm(x, a_kappa, b_kappa)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

a_omega = -3
b_omega = 1
x <- seq(0, 1, length = 100)
hx <- dlnorm(x, a_omega, b_omega)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")
```

```
l_lambda = -1
u_lambda = 0.99
x <- seq(-1, 1, length = 100)

hx <- dunif(x, l_lambda, u_lambda)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

a_theta = 4
b_theta = 1
x <- seq(0, 100, length = 100)
hx <- dlnorm(x, a_theta, b_theta)
plot(x, hx, type = "l", lty = 1, xlab = "x value",
     ylab = "Density", main = "Prior")

est = estgtp(X$X,
          skappa = exp(a_kappa + ((b_kappa ^ 2) / 2)) / 100,
          somega = exp(a_omega + ((b_omega ^ 2) / 2)) / 100,
          dlambda = 0.01,
          stheta = exp(a_theta + ((b_theta ^ 2) / 2)) / 100, smove = 0.1,
          a_kappa = a_kappa, b_kappa = b_kappa,
          a_omega = a_omega, b_omega = b_omega,
          l_lambda = l_lambda, u_lambda = u_lambda,
          a_theta = a_theta, b_theta = b_theta,
          iter = 50, plot.step = 50, save.step = 1e9,
          filename = "")

discard = 10
step = 10

result = estgtpr(est, discard, step)
```

`estinternsp` *Estimation of interaction Neyman-Scott point process using auxiliary variable algorithm into Markov chain Monte Carlo.*

Description

The Bayesian MCMC estimation of parameters for interaction Neyman-Scott point process using auxiliary variable algorithm into Markov chain Monte Carlo.

Usage

```
estinternsp(
  X,
  control,
  x_left,
  x_right,
  y_bottom,
```

```
  y_top,
  W_dil,
  AreaW,
  AreaMRW,
  radius
)
```

Arguments

X	observed point pattern in the ppp format of the **spatstat** package.
control	list specifying various tuning constants for the MCMC estimation. See also Details.
x_left	vector describing the observation window, contains the lower x-coordinate of the corners of each rectangle.
x_right	vector describing the observation window, contains the higher x-coordinate of the corners of each rectangle.
y_bottom	vector describing the observation window, contains the smaller y-coordinate of the corners of each rectangle.
y_top	vector describing the observation window, contains the higher y-coordinate of the corners of each rectangle.
W_dil	observation window dilated by the assumed maximal cluster radius.
AreaW	area of a window.
AreaMRW	area of a dilated window.
radius	The radius of dilation, passed as an argument to dilation.owin to create a dilated window.

Details

Observation window and its dilation

The observation window must be provided as the union of aligned rectangles, aligned with the coordinate axes. This, however, allows the analysis of point patterns observed in rather irregular regions by approximating the region by a union of aligned rectangles. The structure of the vectors *x_left, x_right, y_bottom* and *y_top* is such that the first rectangle is constructed using the function owin from the spatstat package as `owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))`, and similarly for the other rectangles. Naturally, a rectangular window can be used and in such a case the vectors *x_left* to *y_top* each contain a single element.

control

The control list must contain the following elements: ***NStep*** (the required number of MCMC iterations to be performed), ***BurnIn*** (burn-in, how many iterations at the beginning of the chain will be disregarded when computing the resulting estimates), ***SamplingFreq*** (sampling frequency for estimating the posterior distributions). Additionally, the hyperparameters for the prior distributions should be given, see below.

Prior distributions and hyperparameters

The prior distribution for each parameter (*alpha*, *omega*, *kappa*, *theta1*, *theta2*) is normal with respective `mean=Prior_<parameter>_mean` and `SD=Prior_<parameter>_SD`. During update, the

Lower Bound for each parameter is <parameter>_LB and the Upper Bound is <parameter>_UB. *alpha* controls the expected number of occurrences or events around specified focal points or regions, *omega* controls the width of cluster events activity and *kappa* controls the overall intensity of the parent process. *theta1* controls the shape of the interaction function and *theta2* controls the shape of the interaction function and provides the location of the peak value.

Value

The output of the function is given by the list containing the parameter estimates along with the 2.5% and 97.5% confidence interval of the posterior distributions, cluster centers, *control*, W_dil and parameters.

References

Park, J., Chang, W., & Choi, B. (2022). An interaction Neyman-Scott point process model for coronavirus disease-19. *Spatial Statistics*, 47, 100561.

See Also

print.output_estinternsp

Examples

```
library(spatstat)
# library(spatstat.geom)
library(fields)
library(mvtnorm)
library(binspp)

# Generate example window
W <- owin(xrange = c(0, 100), yrange = c(0, 50)) # example window (rectangle)
radius = 2
W_dil = dilation.owin(W, radius)
AreaW <- area(W)
AreaMRW <- area(W_dil)

x_left = W$xrange[1]
x_right = W$xrange[2]
y_bottom = W$yrange[1]
y_top = W$yrange[2]

# True parameters
alpha <- 15
omega <- 1.5
kappa <- 0.001
theta1 <- 3
theta2 <- 10

# Generate parents process
CCC <- rpoispp(kappa, win = W_dil)
CC <- t(rbind(CCC$x,CCC$y))
CCdist <- rdist(CC)
```

```
r <- binspp::coeff(c(theta1,theta2))
r1 <- r[1]; r2 <- r[2]; t1 <- theta1; t2 <- theta2; t3 <- 0.5; R <- 0;
rho0sum <- rep(0,dim(CC)[1])               # row sum of rho matrix
res <- binspp::pCClik2(c(theta1,theta2), CC)
rho0sum <- res$rhosum
likelihoodprev <- res$likelihood

# this is just for example, for a real estimate
# use value of at least niter = 10000
CC.true <- AuxVarGen(kappa, c(theta1,theta2), likelihoodprev, rho0sum, CC,
    AreaMRW, W_dil, 100)

# Generate offsprings given parents
gaus <- function(n, omega) {
  matrix(rnorm(2 * n, mean=0, sd=omega), ncol=2)
}
parents <- ppp(CC.true[[1]][,1],CC.true[[1]][,2],window=W)
np <- npoints(parents)

csize <- qpois(runif(np, min = dpois(0, alpha)), alpha)
noff <- sum(csize)
xparent <- parents$x
yparent <- parents$y
x0 <- rep.int(xparent, csize)
y0 <- rep.int(yparent, csize)
dd <- gaus(noff, omega)
xy <- xy.coords(dd)
dx <- xy$x; dy <- xy$y

xoff <- x0 + dx; yoff <- y0 + dy
result <- ppp(xoff, yoff, window = W_dil, check = FALSE, marks = NULL)
X <- cbind(result$x,result$y) # Generate example data

# this is just for example, for a real estimate
# use values of at least NStep = 20000 and BurnIn = 5000
control = list(NStep = 50, BurnIn = 25, SamplingFreq = 10,
   Prior_alpha_mean = 15, Prior_alpha_SD = 2, alpha_LB = 10, alpha_UB = 20,
   Prior_omega_mean = 1.5, Prior_omega_SD = 0.2, omega_LB = 1, omega_UB = 2,
   Prior_kappa_mean = 0.001, Prior_kappa_SD = 0.0001, kappa_LB = 0.0005,
                             kappa_UB = 0.002,
   Prior_theta1_mean = 3, Prior_theta1_SD = 0.2, theta1_LB = 2.5,
                             theta1_UB = 3.5,
   Prior_theta2_mean = 10, Prior_theta2_SD = 1, theta2_LB = 8,
                             theta2_UB = 12)

Output = estinternsp(X, control,
                     x_left, x_right, y_bottom, y_top,
                     W_dil,
                     AreaW, AreaMRW, radius)
```

`estintp` *Estimation of Thomas-type cluster point process with complex inhomogeneities*

Description

The Bayesian MCMC estimation of parameters for Thomas-type cluster point process with inhomogeneity is performed in any of the following parts: (i) distribution of parent points, (ii) mean number of points in a cluster, (iii) cluster spread. The process is observed in the observation window W which is a union of aligned rectangles, aligned with the coordinate axes. The inhomogeneities are described through a parametric model depending on covariates. The estimation algorithm is described in Dvořák, Remeš, Beránek & Mrkvička (2022) (doi:10.48550/arXiv.2205.07946).

Usage

```
estintp(
  X,
  control,
  x_left = NULL,
  x_right = NULL,
  y_bottom = NULL,
  y_top = NULL,
  W = NULL,
  W_dil,
  z_beta = NULL,
  z_alpha = NULL,
  z_omega = NULL,
  credibility.level = 0.95,
  verbose = TRUE
)
```

Arguments

`X`	observed point pattern in the `spatstat.geom::ppp()` format of the **spatstat** package.
`control`	list specifying various tuning constants for the MCMC estimation. See also Details.
`x_left`	vector describing the observation window, contains the lower x-coordinate of the corners of each rectangle.
`x_right`	vector describing the observation window, contains the higher x-coordinate of the corners of each rectangle.
`y_bottom`	vector describing the observation window, contains the smaller y-coordinate of the corners of each rectangle.
`y_top`	vector describing the observation window, contains the higher y-coordinate of the corners of each rectangle.

W	a rectangular observation window can be provided directly using this argument; in this case, the parameters x_left to y_top are determined automatically.
W_dil	the observation window dilated by the assumed maximal cluster radius.
z_beta	list of covariates describing the intensity function of the parent process, each covariate being a pixel image as used in the **spatstat** package.
z_alpha	list of covariates describing the location-dependent mean number of points in a cluster, each covariate being a pixel image as used in the **spatstat** package.
z_omega	list of covariates describing the location-dependent scale of a cluster, each covariate being a pixel image as used in the **spatstat** package.
credibility.level	credibility level of the credible intervals to be estimated for the posterior distributions; defaults to 0.95.
verbose	logical (TRUE or FALSE). For suppressing information messages to the console set value to FALSE. Defaults to TRUE.

Value

The output of the function is given by the list containing the parameter estimates along with the 2.5% and 97.5% quantiles of the posterior distributions. Also, several auxiliary objects are included in the list which are needed for the `print.output_estintp()` and `plot.output_estintp()` functions.

Details

Parametric model:

The model for the intensity function of the parent process is the following: $f(u) = kappa * exp(beta_1 * z\_beta_1(u) + \ldots + beta_k * z\_beta_k(u))$, where $(kappa, beta_1, \ldots, beta_k)$ is the vector of parameters and $z\_beta = (z\_beta_1, \ldots, z\_beta_k)$ is the list of covariates. Note that choosing $k = 0$ is acceptable, resulting in a homogeneous distribution of parents. In such a case *z_beta* must be an empty list or NULL. Furthermore, the list z_beta must contain named covariates in order to properly function with the function `spatstat.model::ppm()` from the **spatstat** package which is used in the first step to estimate the parameters $(kappa, beta_1, \ldots, beta_k)$. Note that due to identifiability issues the covariate lists *z_beta* and *z_alpha* must be disjoint.

The model for the mean number of points in a cluster corresponding to the parent at location u is the following: $g(u) = exp(alpha + alpha_1 * z\_alpha_1(u) + \ldots + alpha_l * z\_alpha_l(u))$, where $(alpha, alpha_1, \ldots, alpha_l)$ is the vector of parameters and $z\_alpha = (z\_alpha_1, \ldots, z\_alpha_l)$ is the list of covariates. Note that choosing $l = 0$ is acceptable, resulting in a constant model. In such a case *z_alpha* must be an empty list or NULL. Note that due to identifiability issues the covariate lists *z_beta* and *z_alpha* must be disjoint. A warning is issued when this is not the case.

The model for the scale of a cluster corresponding to the parent at location u is the following: $h(u) = exp(omega + omega_1 * z\_omega_1(u) + \ldots + omega_m * z\_omega_m(u))$, where $(omega, omega_1, \ldots, omega_m)$ is the vector of parameters and $z\_omega = (z\_omega_1, \ldots, z\_omega_m)$ is the list of covariates. Note that choosing $m = 0$ is acceptable, resulting in a constant model. In such a case *z_omega* must be an empty list or NULL.

Observation window and its dilation:

The observation window must be provided as the union of aligned rectangles, aligned with the coordinate axes. This, however, allows the analysis of point patterns observed in rather irregular

regions by approximating the region by a union of aligned rectangles. The structure of the vectors *x_left, x_right, y_bottom* and *y_top* is such that the first rectangle is constructed using the function `spatstat.geom::owin()` from the **spatstat** package as `owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))`, and similarly for the other rectangles. A rectangular window can be provided directly in the argument *W* and in such case the vectors *x_left* to *y_top* are determined automatically.

Covariates:

The covariates must be provided as pixel images of the class `spatstat.geom::im()` used in the **spatstat** package. It is recommended that all the covariates have the same pixel resolution. However, it is necessary that all the covariates in the list *z_beta* have the same resolution, all the covariates in the list *z_alpha* have the same resolution and all the covariates in the list *z_omega* have the same distribution. The covariates must be provided in the dilated observation window *W_dil*, with NA values at pixels lying outside *W_dil*.

Control:

The control list must contain the following elements: *NStep* (the required number of MCMC iterations to be performed), *BurnIn* (burn-in, how many iterations at the beginning of the chain will be disregarded when computing the resulting estimates – note that this choice can be updated after the computation without re-running the chain, see the function `re_estimate()`), *SamplingFreq* (sampling frequency for estimating the posterior distributions). Additionally, the hyperparameters for the prior distributions should be given, see below. Note that some default values for the hyperparameters are provided but it is **strongly encouraged** that the hyperparameter values are given by the user, based on the actual knowledge of the problem at hand.

Prior distributions and hyperparameters:

The prior distribution for *alpha* is normal with `mean = Prior_alpha_mean` and `SD = Prior_alpha_SD`.

The prior distribution for the vector $(alpha_1, \ldots, alpha_l)$ is centered normal with diagonal variance matrix and the vector of `SDs = Prior_alphavec_SD`.

The prior distribution for *omega* is normal with `mean = Prior_omega_mean` and `SD = Prior_omega_SD`.

The prior distribution for the vector $(omega_1, \ldots, omega_m)$ is centered normal with diagonal variance matrix and the vector of `SDs = Prior_omegavec_SD`.

The hyperparameters should be provided in the control list. However, the following default choices are applied if the hyperparameter values are not provided by user or are given as NULL: `Prior_alpha_mean = 3`, `Prior_alpha_SD = 2`, `Prior_omega_mean = log(sqrt(area(W) / 20))`, `Prior_omega_SD = log(3 + sqrt(area(W) / 40))`, `Prior_alphavec_SD[i] = 2 / max(z_alpha_i)`, `Prior_omegavec_SD[i] = 2 / max(z_omega_i) * log(3 + sqrt(area(W) / 20))`.

Output:

The output of the function is given by the list containing the parameter estimates along with the credible intervals of the posterior distributions. Also, several auxiliary objects are included in the list which are needed for the `print.output_estintp()` and `plot.output_estintp()` functions.

Examples

```
library(spatstat)
```

```
library(spatstat.geom)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# User-specified hyperparameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5,
    Prior_alpha_mean = 3, Prior_alpha_SD = 2, Prior_omega_mean = 5.5,
    Prior_omega_SD = 5, Prior_alphavec_SD = c(4.25, 0.012),
    Prior_omegavec_SD = c(0.18,0.009))

# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)

# Text output + series of figures:
print(Output)
plot(Output)

# Simulate from the fitted model:
simulate(Output)

# Access the raw outputs of the MCMC run:
rawMCMCoutput(Output)
```

---

first_step *Estimate the first-order inhomogeneity*

Description

For exploratory purposes it may be useful to perform the first step of the analysis only, to investigate the dependence of the intensity function of the parent process on given covariates, without running the MCMC chain.

Usage

```
first_step(X, z_beta, W_dil, plot = TRUE)
```

Arguments

X	observed point pattern in the spatstat.geom::ppp() format of the **spatstat** package.
z_beta	list of covariates describing the intensity function of the parent process, each covariate being a pixel image as used in the **spatstat** package.
W_dil	the observation window dilated by the assumed maximal cluster radius.
plot	logical, should the estimates intensity function of the parent process be plotted?

Details

The calling the spatstat.model::ppm() function from the **spatstat** package, with some additional computations useful when preparing the run of the MCMC chain, is mainly performed in this function. The function also contains a simple way to plot the estimated intensity function of the parent process.

Value

List containing the output of the spatstat.model::ppm() function from the **spatstat** package, along with some auxiliary objects useful for running the MCMC chain.

Examples

```
library(spatstat)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2){
  for (i in 2:length(x_left)){
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}
```

```
# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Estimating the intensity function of the parent process:
aux <- first_step(X = X, z_beta = z_beta, W_dil = W_dil, plot = TRUE)
aux
```

---

pCClik2 *Evaluate unnormalized likelihood for auxiliary variable*

Description

Calculates the unnormalized likelihood for an auxiliary variable by evaluating pairwise interaction between points. The interaction thresholds are derived from the input theta vector.

Usage

```
pCClik2(thetaprop, CC)
```

Arguments

thetaprop	A vector of theta1 and theta2.
CC	A coordinate matrix of points.

Value

A list of the computed likelihood and a row vector of summed interaction between points.

---

plot.output_estintp *Graphical output describing the posterior distributions*

Description

A graphical representation of the posterior distributions in terms of histograms and trace plots.

Usage

```
## S3 method for class 'output_estintp'
plot(x, ...)
```

Arguments

x	list, output of the main function estintp().
...	further arguments passed from generic print function.

Details

If the covariate list *z_beta* was non-empty, the estimated intensity function of the parent process is plotted. Then, the estimated surface representing the location dependent mean number of points in a cluster is plotted, and similarly, the estimated surface representing the location dependent scale of clusters is plotted.
After that, histograms of the sample posterior distributions of the individual parameters are plotted, together with the histograms of p-values giving significance of the individual covariates in *z_beta* with respect to the population of parent points.
Then, the trace plots for individual model parameters are plotted, with highlighted sample median (full red line) and sample 2.5% and 97.5% quantiles (dashed red lines), and similarly for the p-values giving significance of the individual covariates in *z_beta* with respect to the population of parent points.
Additionally, the following graphs are also plotted:

- trace plot for the log-likelihood of the model,
- trace plot for the number of parent points,
- trace plot for the probability of accepting proposed updates of $(alpha, alpha_1, \ldots, alpha_l)$,
- trace plot for the fraction of accepted updates of $alpha, alpha_1, \ldots, alpha_l$ in the last 1000 iterations,
- trace plot for the probability of accepting proposed updates of $omega, omega_1, \ldots, omega_m$,
- trace plot for the fraction of accepted updates of $omega, omega_1, \ldots, omega_m$ in the last 1000 iterations,
- trace plot for the fraction of accepted updates of parent points in the last 1000 iterations.

Value

Series of plots providing a graphical representation of the posterior distributions in terms of histograms and trace plots.

Examples

```
library(spatstat)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
```

```
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Default parameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5)


# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)

# Text output + series of figures:
print(Output)
plot(Output)
```

---

plot_conn *plot_conn*

Description

Auxiliary function to plot partial results during evaluation of estgtp.

Usage

```
plot_conn(X, C)
```

Arguments

X	The input set from the estgtp function
C	Prepared parameter from the estgtp function

Details

Auxiliary function which plots next step of partial results during calculation of the estgtp function.

Examples

```
library(spatstat)
kappa = 10
omega = .1
lambda= .5
theta = 10
```

```
X = rgtp(kappa, omega, lambda, theta, win = owin(c(0, 1), c(0, 1)))
plot_conn(X$X, X$C)
```

`print.output_estinternsp` *Text output describing the posterior distributions*

Description

Output printing function of interaction neyman-scott process. It prints the estimated values and the posterior confidence intervals for each parameter.

Usage

```
## S3 method for class 'output_estinternsp'
print(x, ...)
```

Arguments

`x`	Output of the main function `estinternsp`.
`...`	further arguments passed from generic print function.

Value

Text output summarizing the posterior distributions for each parameter.

See Also

`estinternsp`

Examples

```
library(spatstat)
# library(spatstat.geom)
library(fields)
library(mvtnorm)
library(binspp)

# Generate example window
W <- owin(xrange = c(0, 100), yrange = c(0, 50)) # example window (rectangle)
radius = 2
W_dil = dilation.owin(W, radius)
AreaW <- area(W)
AreaMRW <- area(W_dil)

x_left = W$xrange[1]
x_right = W$xrange[2]
```

```
y_bottom = W$yrange[1]
y_top = W$yrange[2]

# True parameters
alpha <- 15
omega <- 1.5
kappa <- 0.001
theta1 <- 3
theta2 <- 10

# Generate parents process
CCC <- rpoispp(kappa, win = W_dil)
CC <- t(rbind(CCC$x,CCC$y))
CCdist <- rdist(CC)
r <- binspp::coeff(c(theta1,theta2))
r1 <- r[1]; r2 <- r[2]; t1 <- theta1; t2 <- theta2; t3 <- 0.5; R <- 0;
rho0sum <- rep(0,dim(CC)[1])                # row sum of rho matrix
res <- binspp::pCClik2(c(theta1,theta2), CC)
rho0sum <- res$rhosum
likelihoodprev <- res$likelihood

# this is just for example, for a real estimate
# use value of at least niter = 10000
CC.true <- AuxVarGen(kappa, c(theta1,theta2), likelihoodprev, rho0sum, CC,
    AreaMRW, W_dil, 100)

# Generate offsprings given parents
gaus <- function(n, omega) {
  matrix(rnorm(2 * n, mean=0, sd=omega), ncol=2)
}
parents <- ppp(CC.true[[1]][,1],CC.true[[1]][,2],window=W)
np <- npoints(parents)

csize <- qpois(runif(np, min = dpois(0, alpha)), alpha)
noff <- sum(csize)
xparent <- parents$x
yparent <- parents$y
x0 <- rep.int(xparent, csize)
y0 <- rep.int(yparent, csize)
dd <- gaus(noff, omega)
xy <- xy.coords(dd)
dx <- xy$x; dy <- xy$y

xoff <- x0 + dx; yoff <- y0 + dy
result <- ppp(xoff, yoff, window = W_dil, check = FALSE, marks = NULL)
X <- cbind(result$x,result$y) # Generate example data

# this is just for example, for a real estimate
# use values of at least NStep = 20000 and BurnIn = 5000
control = list(NStep = 50, BurnIn = 25, SamplingFreq = 10,
   Prior_alpha_mean = 15, Prior_alpha_SD = 2, alpha_LB = 10, alpha_UB = 20,
   Prior_omega_mean = 1.5, Prior_omega_SD = 0.2, omega_LB = 1, omega_UB = 2,
   Prior_kappa_mean = 0.001, Prior_kappa_SD = 0.0001, kappa_LB = 0.0005,
```

```
                              kappa_UB = 0.002,
    Prior_theta1_mean = 3, Prior_theta1_SD = 0.2, theta1_LB = 2.5,
                              theta1_UB = 3.5,
    Prior_theta2_mean = 10, Prior_theta2_SD = 1, theta2_LB = 8,
                              theta2_UB = 12)

Output = estinternsp(X, control,
                     x_left, x_right, y_bottom, y_top,
                     W_dil,
                     AreaW, AreaMRW, radius)
print(Output)
```

`print.output_estintp` *Text output describing the posterior distributions*

Description

The summaries of the posterior distributions in the text form are provided.

Usage

```
## S3 method for class 'output_estintp'
print(x, ...)
```

Arguments

`x`	list, output of the main function `estintp()`.
`...`	further arguments passed from generic print function.

Details

The parameter estimates (sample medians from the empirical posterior distributions) and the corresponding credible intervals are printed.
Additionally, during the run of the MCMC chain the significance of the covariates in the list *z_beta* with respect to the current population of parent points is repeatedly tested. This function prints the median of the series of p-values obtained in this way for each covariate, together with the corresponding credible interval.

Value

Text output summarizing the posterior distributions.

Examples

```
library(spatstat)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Default parameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5)


# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)


# Text output
print(Output)
```

`rawMCMCoutput` *Obtaining the raw MCMC output*

Description

Access directly the raw values recorded during the run of the MCMC algorithm.

Usage

```
rawMCMCoutput(Output)
```

Arguments

`Output`	list, output of the main function `estintp()`.

Details

This function provides access to the intermediate values recorded during the run of the MCMC algorithm. These can be used to approximate the posterior distributions of various model parameters or construct various diagnostic plots. No burn-in period is considered here, i.e. the function returns the values recorded during the whole run of the Markov chain.

Value

List with named components, providing the vectors of values recorded during the run of the MCMC algorithm.

Examples

```
library(spatstat)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Default parameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5)


# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)
```

```
# Access the raw outputs of the chain:
rawMCMCoutput(Output)
```

`re_estimate` *Re-estimate the posterior distributions with a different burn-in or a different credibility level.*

Description

After running the MCMC chain for the given number of steps, the trace plots may indicate that a too small value of burn-in was used in the first place. This function enables re-estimating the posterior distributions with a different value of burn-in, without the need to run the MCMC chain again. Similarly, it allows determining the credible intervals of the posterior distributions with a different credibility level.

Usage

```
re_estimate(Output, BurnIn = 0, credibility.level = NULL)
```

Arguments

`Output`	list, output of the main function estintp.
`BurnIn`	new value of burn-in.
`credibility.level`	new value of the credibility level.

Details

The output of the main function binspp contains all the intermediate states of the chain (sampled with the required frequency) no matter what the original value of burn-in was. This enables a simple and quick re-estimation of the posterior distributions with either higher or lower value of burn-in than the one used originally. Similarly, this function can be used to determine the credible intervals of the posterior distributions with a different credibility level. The output of this function has the same structure as the output of the main function `estintp()`.

Value

List containing the parameter estimates along with the credible intervals of the posterior distributions, along with auxiliary objects needed for printing and plotting the outputs.

Examples

```
library(spatstat)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Default parameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5)

# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)

# Text output + series of figures:
print(Output)
plot(Output)

# Recompute the outputs when another value of burn-in is desired,
# without running the chain again:
Out2 <- re_estimate(Output, BurnIn = 80)
print(Out2)
plot(Out2)
```

---

rgtp *Simulation of generalized Thomas process*

Description

Simulation of generalized Thomas process.

Usage

```
rgtp(
  kappa,
  omega,
  lambda,
  theta,
  win = owin(c(0, 1), c(0, 1)),
  nsim = 1,
  expand = 4 * omega,
  C = NULL
)
```

Arguments

kappa	intensity of cluster centers.
omega	standard deviation of normal distribution specifying the clusters spread.
lambda	parameter of generalised Poisson distribution controlling over or under dispersion.
theta	parameter of generalised Poisson distribution controlling the mean number of points in a cluster.
win	window in which to simulate the pattern. An object in the spatstat.geom::owin() format of the **spatstat** package.
nsim	number of simulations.
expand	the size of expansion of window to simulate the centers of clusters.
C	.

Value

A list(X, C), where *X* is Generalized Thomas process, and *C* is Process of cluster centers for Generalized Thomas process.

Examples

```
library(spatstat.geom)
kappa = 10
omega = .1
lambda= .5
theta = 10

X = rgtp(kappa, omega, lambda, theta, win = owin(c(0, 1), c(0, 1)))
plot(X$X)
plot(X$C)
```

`rThomasInhom` *Simulate a realization of Thomas-type cluster point process with complex inhomogeneities*

Description

The means to simulate realizations from the Thomas-type cluster point process with complex inhomogeneities are provided.

Usage

```
rThomasInhom(
  kappa,
  alpha,
  omega,
  W,
  W_dil,
  betavec = NULL,
  alphavec = NULL,
  omegavec = NULL,
  z_beta = NULL,
  z_alpha = NULL,
  z_omega = NULL
)
```

Arguments

`kappa`	intensity or intensity function of the parent process, scalar or pixel image object of class `spatstat.geom::im()` from the **spatstat** package.
`alpha`	scalar, influences the mean number of points in individual clusters, see Details.
`omega`	scalar, influences the spread of individual clusters, see Details.
`W`	the observation window where the realization is to be generated, in the `spatstat.geom::owin()` format of the **spatstat** package.
`W_dil`	the observation window dilated by the assumed maximal cluster radius, as a binary mask with the same resolution as the covariates.
`betavec`	vector of parameters describing the dependence of the intensity function of the parent process on covariates in the list *z_beta*.
`alphavec`	vector of parameters describing the dependence of the mean number of points in a cluster on covariates in the list *z_alpha*.
`omegavec`	vector of parameters describing the dependence of the spread of the clusters on covariates in the list *z_omega*.
`z_beta`	list of covariates describing the intensity function of the parent process, each covariate being a pixel image as used in the **spatstat** package.
`z_alpha`	list of covariates describing the location-dependent mean number of points in a cluster, each covariate being a pixel image as used in the **spatstat** package.

z_omega — list of covariates describing the location-dependent scale of a cluster, each covariate being a pixel image as used in the **spatstat** package.

Details

A realization of a Thomas-type cluster point process model is produced with possible inhomogeneity (described by covariates) in any or all of the following model components: intensity function of the parent process, mean number of points in a cluster, scale of the clusters. Model parametrization is described in the documentation to the function `estintp()`. The parent process is generated using the function `spatstat.random::rpoispp()` from the **spatstat.random** package in the dilated observation window *W_dil* to avoid edge-effects. The offspring points are generated directly from the appropriate normal distributions. The resulting point pattern is eventually clipped to the smaller observation window *W*.

Value

A planar point pattern, object of the type `spatstat.geom::ppp()` used in the **spatstat** package.

Examples

```
library(spatstat)
# Unit square observation window:
W <- owin()

# Dilation of the observation window:
W_dil <- dilation(W, 0.1)
W_dil <- as.mask(W_dil)

# Define covariates:
f1 <- function(x, y) { x }
f2 <- function(x, y) { y }
f3 <- function(x, y) { 1 - (y - 0.5) ^ 2 }
cov1 <- as.im(f1, W = W_dil)
cov2 <- as.im(f2, W = W_dil)
cov3 <- as.im(f3, W = W_dil)


# Stationary Thomas process:
X <- rThomasInhom(kappa = 50, alpha = log(10), omega = log(0.01),
       W = W, W_dil = W_dil)
plot(X)


# Thomas-type cluster process with inhomogeneity in all model components:
X <- rThomasInhom(kappa = 10, betavec = c(1), z_beta = list(cov1),
            alpha = log(10), alphavec = c(1), z_alpha = list(cov2),
            omega = log(0.01), omegavec = c(1), z_omega = list(cov3),
            W = W, W_dil = W_dil)
plot(X)
```

simulate.output_estintp *Simulation from the fitted model*

Description

One or more point patterns are simulated using the point estimates provided by the MCMC output.

Usage

```
## S3 method for class 'output_estintp'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	list, output of the main function estintp().
nsim	number of patterns to be simulated.
seed	a single value, interpreted as an integer, or NULL.
...	additional optional arguments.

Details

A given number of point patterns is simulated from the fitted model. The point pattern used for estimation determines the observation window and the covariates to be used. Point estimates from the output of the MCMC run are used as parameter values.

Value

Either a single point pattern or a list of point patterns.

Examples

```
library(spatstat)
library(stats)
# Prepare the dataset:
X <- trees_N4
x_left <- x_left_N4
x_right <- x_right_N4
y_bottom <- y_bottom_N4
y_top <- y_top_N4

z_beta <- list(refor = cov_refor, slope = cov_slope)
z_alpha <- list(tmi = cov_tmi, tdensity = cov_tdensity)
z_omega <- list(slope = cov_slope, reserv = cov_reserv)

# Determine the union of rectangles:
W <- owin(c(x_left[1], x_right[1]), c(y_bottom[1], y_top[1]))
if (length(x_left) >= 2) {
```

```
  for (i in 2:length(x_left)) {
    W2 <- owin(c(x_left[i], x_right[i]), c(y_bottom[i], y_top[i]))
    W <- union.owin(W, W2)
  }
}

# Dilated observation window:
W_dil <- dilation.owin(W, 100)


# Default parameters for prior distributions:
control <- list(NStep = 100, BurnIn = 20, SamplingFreq = 5)


# MCMC estimation:
Output <- estintp(X = X, control = control, x_left = x_left, x_right = x_right,
    y_bottom = y_bottom, y_top = y_top, W_dil = W_dil, z_beta = z_beta,
    z_alpha = z_alpha, z_omega = z_omega, verbose = FALSE)


# Simulation from the fitted model:
pattern <- simulate(Output)
plot(pattern)
```

`trees_N4` *Spanish oak trees*

Description

The oak trees dataset sampled in 2009 in region consisting of 5 rectangles.

Usage

```
trees_N4
```

Format

A list with columns:

window A list of region window definition.

n Number of oak trees in the region.

x Array of x coordinates of oak trees.

y Array of y coordinates of oak trees.

Details

The data contains point pattern of trees, 5 covariates (refor, reserve, slope, tdensity, tmi) and 4 vectors (x_left, x_right, y_bottom, y_top) of corners of rectangles forming the observation window.

Source

Jesús Fernández-Habas, Pilar Fernández-Rebollo, Mónica Rivas Casado, Alma María García Moreno, Begoña Abellanas. Spatio-temporal analysis of oak decline process in open woodlands: A case study in SW Spain, Journal of Environmental Management, 248, 2019, 109308, doi:10.1016/j.jenvman.2019.109308.

Examples

```
plot(trees_N4)
```

`x_left_N4` *Left horizontal corners for trees_N4 dataset*

Description

The vector of left horizontal corners of rectangles forming observation window for trees_N4 dataset.

Usage

```
x_left_N4
```

Format

An object of class `numeric` of length 5.

`x_right_N4` *Right horizontal corners for trees_N4 dataset*

Description

The vector of right horizontal corners of rectangles forming observation window for trees_N4 dataset.

Usage

```
x_right_N4
```

Format

An object of class `numeric` of length 5.

y_bottom_N4 *Bottom vertical corners for trees_N4 dataset*

Description

The vector of bottom vertical corners of rectangles forming observation window for trees_N4 dataset.

Usage

```
y_bottom_N4
```

Format

An object of class `numeric` of length 5.

y_top_N4 *Vertical corners for trees_N4 dataset*

Description

The vector of top vertical corners of rectangles forming observation window for trees_N4 dataset.

Usage

```
y_top_N4
```

Format

An object of class `numeric` of length 5.

Index