Package: SepTest (via r-universe)

November 4, 2026

Title Tests for First-Order Separability in Spatio-Temporal Point Processes

Version 0.0.1

Date 2025-10-25

Author Mohammad Ghorbani [author, creator], Nafiseh Vafaei [author]

Maintainer Mohammad Ghorbani <mohammad.ghorbani@slu.se>

Description Provides statistical tools for testing first-order separability in spatio-temporal point processes, that is, whether the spatio-temporal intensity function can be expressed as a product of spatial and temporal components. The package implements several hypothesis tests, including exact and asymptotic methods for Poisson and non-Poisson processes. Methods include global envelope tests, chi-squared-type statistics, and a novel Hilbert-Schmidt independence criterion (HSIC) test with block or pure permutations. Simulation studies and real-world examples, including the 2001 UK foot-and-mouth disease outbreak data, illustrate its utility. The package encompasses all simulation studies and applications presented in Ghorbani et al. (2021, 2025).

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports combinat, dHSIC, fields, GET, ggplot2, graphics, MASS, patchwork, reshape2, scatterplot3d, spatstat.explore, spatstat.geom, spatstat.model, splancs, stats

Suggests testthat (>= 3.0.0), rmarkdown, pkgdown, knitr, spatstat, stpp, RColorBrewer

BugReports https://github.com/mghorbani01/SepTest/issues

URL https://github.com/mghorbani01/SepTest

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev

Repository https://mghorbani01.r-universe.dev

Date/Publication 2025-12-26 02:15:45 UTC

RemoteUrl https://github.com/mghorbani01/septest

RemoteRef HEAD

RemoteSha 5f2b4a79ec086ec0fe68e97547a54d30c0756d8e

Contents

block.permut . . . 2
calc.bandwidths.and.edgecorr . . . 4
check.args . . . 5
chi2.test . . . 6
chisq.test.stPP . . . 8
dHS.test . . . 10
estimate.intensity.pixel . . . 12
estimate.intensity.point . . . 13
estimate.st.intensity . . . 14
Gauss.st.F . . . 16
get.lambda.function . . . 18
get.lambda.max . . . 20
global.envelope.test . . . 22
norm2d . . . 25
norm3d . . . 26
plot_procedures . . . 27
plot_stDPP . . . 29
plot_stlgcp . . . 30
plot_stpp . . . 31
random.shift . . . 33
rstDPP . . . 34
rstLGCPP . . . 37
rstpoispp . . . 38
S.based.functions . . . 40
sim.procedures . . . 42

Index **43**

`block.permut` *Block permutation of the temporal component in a spatio-temporal point pattern*

Description

Permutes the temporal component of a spatio-temporal dataset in a block-wise manner while keeping the spatial coordinates fixed. This is used to generate permuted replicates under the null model of first-order separability.

Usage

```
block.permut(nblocks, X, nperm = 1999)
```

Arguments

nblocks	Integer (>= 2). Number of consecutive temporal blocks after ordering events by time.
X	Numeric matrix or data frame with at least three columns (x, y, t). Each row displays one event. The third column is interpreted as the time coordinate.
nperm	Integer (>= 1). Number of permuted datasets to generate. At most factorial(nblocks) - 1 distinct non-identity block permutations exist.

Details

The function first orders the events by time and partitions the ordered sequence into nblocks consecutive blocks of equal size. The block labels are permuted (excluding the identity permutation), and the time values are reassigned according to the permuted block order.

If nrow(X) is not divisible by nblocks, the last nrow(X) %% nblocks events are not included in the block permutation and are appended unchanged to each permuted dataset.

For details of the block permutation procedure, see the Supplementary Materials in Ghorbani et al. (2025).

sim.procedures covers both pure and block permutation methods.

Value

A list of length min(nperm, factorial(nblocks) - 1). Each element is a matrix with the same number of columns as X; the third column contains the block-permuted time values.

References

Ghorbani, M., Vafaei, N. and Myllymäki, M. (2025). A kernel-based test for the first-order separability of spatio-temporal point processes, *TEST*.

See Also

sim.procedures

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), runif(100, 0, 10))
perms <- block.permut(nblocks = 5, X = X, nperm = 10)
head(perms[[1]], 5)
```

`calc.bandwidths.and.edgecorr` *Compute bandwidths and (Gaussian) edge-correction factors for spatio-temporal kernel intensity estimation*

Description

Computes spatial and temporal bandwidths for kernel-based estimation of the intensity of a spatio-temporal point pattern. The spatial bandwidth is estimated from the spatial coordinates using Diggle's (1985) mean-square error method via `bw.diggle`. The temporal bandwidth is estimated from the time coordinates using the Sheather–Jones direct plug-in method (`bw = "SJ-dpi"`) as implemented in `density`.

Usage

```
calc.bandwidths.and.edgecorr(
  X,
  s.region,
  t.region,
  n.grid,
  epsilon = NULL,
  delta = NULL
)
```

Arguments

`X`	A numeric matrix or data frame with at least three columns giving the x-, y-, and time coordinates t of observed spatio-temporal events. Each row corresponds to one event.
`s.region`	A numeric matrix with two columns giving the vertices of the polygonal spatial observation window in order (the first vertex need not be repeated at the end).
`t.region`	A numeric vector of length 2 giving the temporal observation window `c(tmin, tmax)` with `tmin < tmax`.
`n.grid`	An integer vector of length 3 giving the number of grid cells in the x, y, and time dimensions. Only `n.grid[3]` is used when estimating the temporal bandwidth.
`epsilon`	Optional positive numeric. Spatial bandwidth. If `NULL`, estimated using `bw.diggle`.
`delta`	Optional positive numeric. Temporal bandwidth. If `NULL`, estimated using `density(..., bw = "SJ-dpi")`.

Details

Edge-correction factors are computed for Gaussian kernels as the kernel mass inside the observation window: $c(x) = \int_W K_h(u - x)\, du$. The temporal correction is computed exactly on `t.region`. The spatial correction is computed using a **bounding-box approximation** of the polygonal spatial window (i.e., `W` is replaced by its bounding rectangle).

The spatial window is converted to an owin object, and the spatial bandwidth is estimated using bw.diggle on the corresponding ppp object. The temporal bandwidth is taken as the bandwidth selected by density with bw = "SJ-dpi" over the interval t.region.

The returned edge-correction factors are kernel masses inside the window. If you use them for intensity estimation with edge correction, typical usage is to divide by these factors.

Value

A list with components:

bw Numeric vector of length 3: c(epsilon, epsilon, delta).

time Numeric vector of length nrow(X) giving temporal edge masses $\int_{tmin}^{tmax} K_\delta(u - t_i)\, du$.

space Numeric vector of length nrow(X) giving spatial edge masses computed on the bounding box of s.region.

References

Baddeley A, Rubak E, Turner R (2015). *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press.

Diggle, P.J. (1985). A Kernel Method for Smoothing Point Process Data. *Journal of the Royal Statistical Society, Series C*, 34, 138–147.

See Also

bw.diggle, density

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), runif(100, 0, 10))
s.region <- matrix(c(0,0, 1,0, 1,1, 0,1), ncol = 2, byrow = TRUE)
t.region <- c(0, 10)
n.grid <- c(25, 25, 20)
res <- calc.bandwidths.and.edgecorr(X, s.region, t.region, n.grid)
str(res)
```

---

check.args *Validate common arguments for spatio-temporal grid-based routines*

Description

Checks the validity of common inputs used by spatio-temporal estimation and simulation routines: event locations X, spatial window specification s.region, temporal window t.region, and grid resolution n.grid.

Usage

```
check.args(X, s.region, t.region, n.grid = c(25L, 25L, 20L))
```

Arguments

`X`	A matrix or data frame with at least three columns giving event coordinates (x, y, t). Only the first three columns are used for validation.
`s.region`	Spatial observation region specification. Either: • a numeric vector of length 4 giving `c(xmin, xmax, ymin, ymax)`, or • a numeric matrix with two columns giving polygon vertices `(x, y)` (at least 3 rows). All values must be finite.
`t.region`	Numeric vector of length 2 giving `c(tmin, tmax)` with `tmin < tmax`. Values must be finite.
`n.grid`	Integer vector of length 3 giving the number of grid cells in the x, y, and time directions. Must be positive.

Details

This function is intended as a lightweight argument checker used internally by multiple functions. It can also be called directly for debugging input issues.

Value

Invisibly returns `NULL`. Called for its side effect of throwing an error if inputs are invalid.

Examples

```
X <- cbind(runif(100), runif(100), runif(100, 0, 10))
s.region <- matrix(c(0,0, 1,0, 1,1, 0,1), ncol = 2, byrow = TRUE)
t.region <- c(0, 10)
check.args(X, s.region, t.region, n.grid = c(25, 25, 20))
```

`chi2.test` *Chi-squared test for first-order separability of a spatio-temporal point process*

Description

Performs a chi-squared test for testing first-order separability of a spatio-temporal point process. Two procedures are available:

`"pure_per"` Classical asymptotic chi-squared test of independence on a space–time count table.

`"block_per"` Monte Carlo permutation test based on block-wise permutations of the time component.

Usage

```
chi2.test(
  X,
  sim.procedure = c("pure_per", "block_per"),
  nblocks = 5L,
  nperm = 199L,
  n.time = 2L,
  n.space = 3L,
  t.region = c(0, 1),
  s.region = c(0, 1, 0, 1)
)
```

Arguments

X	A numeric matrix or data frame with at least three columns giving event coordinates (x, y, t).
sim.procedure	Character string specifying the procedure: "pure_per" or "block_per".
nblocks	Integer (>= 2). Number of temporal blocks used for block permutation (only for "block_per").
nperm	Integer (>= 1). Number of Monte Carlo permutations (only for "block_per").
n.time	Integer (>= 2). Number of temporal intervals in the contingency table.
n.space	Integer (>= 2). The spatial domain is partitioned into n.space bins per axis (yielding n.space^2 spatial cells) for the contingency table.
t.region	Numeric vector of length 2 giving the temporal window c(tmin, tmax) with tmin < tmax.
s.region	Spatial window specification. By default, the bounding box c(0, 1, 0, 1) corresponding to c(xmin, xmax, ymin, ymax). Passed to chisq.test.stPP.

Details

The classical procedure ("pure_per") applies a chi-squared test of independence to the n.space^2 by n.time contingency table of counts.

The permutation procedure ("block_per") generates nperm block-permuted datasets under the null using sim.procedures with method = "block", recomputes the chi-squared statistic for each, and returns a Monte Carlo p-value computed as $(1 + \#\{T_i \geq T_{obs}\})/(nperm + 1)$.

Value

Numeric scalar: the p-value of the test.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics and Data Analysis*, **161**, 107245.

Ghorbani, M., Vafaei, N. and Myllymäki, M. (2025). A kernel-based test for the first-order separability of spatio-temporal point processes, *TEST*.

See Also

`chisq.test.stPP`, `sim.procedures`, `block.permut`

Examples

```
set.seed(124)
lambda <- get.lambda.function(N = 200, g = 50, model = 4)
Lmax <- get.lambda.max(N = 200, g = 50, model = 4)
X <- rstpoispp(lambda, Lmax)


# Classical chi-squared test
chi2.test(X, sim.procedure = "pure_per", n.time = 2, n.space = 3)

# Monte Carlo permutation test with blocks
chi2.test(X, sim.procedure = "block_per", nblocks = 5, nperm = 100)
```

`chisq.test.stPP` *Chi-squared test for first-order separability of a spatio-temporal point process*

Description

Performs the classical (asymptotic) chi-squared test of first-order separability by constructing a space–time contingency table of counts and applying a chi-squared test of independence.

Usage

```
chisq.test.stPP(
  X,
  n.space = 2L,
  n.time = 3L,
  s.region = c(0, 1, 0, 1),
  t.region = c(0, 1)
)
```

Arguments

`X`	A numeric matrix or data frame with at least three columns giving event coordinates (x, y, t).
`n.space`	Integer (>= 2). Number of bins per spatial axis. The contingency table has `n.space^2` rows.
`n.time`	Integer (>= 2). Number of temporal bins (columns of the contingency table).
`s.region`	Numeric vector of length 4 giving the spatial bounding box `c(xmin, xmax, ymin, ymax)`. Defaults to the unit box `c(0,1,0,1)`.
`t.region`	Numeric vector of length 2 giving the temporal window `c(tmin, tmax)` with `tmin` < `tmax`. Defaults to `c(0,1)`.

Details

The spatial domain is partitioned into `n.space` bins in each coordinate direction (yielding `n.space^2` spatial cells), and the temporal domain is partitioned into `n.time` intervals. Bin boundaries are defined using empirical quantiles of the observed coordinates, with the first/last boundaries fixed to the provided spatial and temporal windows.

This implementation uses `chisq.test` on the contingency table of space–time counts. If expected counts are very small, the chi-squared approximation may be poor; in that case consider using a Monte Carlo approach (e.g., block permutation) as implemented in `chi2.test`.

Value

A list with components:

chisq_s Numeric scalar. The chi-squared test statistic.

chisq_p Numeric scalar. The p-value of the chi-squared test.

counts Integer matrix of dimension `n.space^2` by `n.time` containing the space–time counts.

Author(s)

Jiří Dvořák <dvorak@karlin.mff.cuni.cz >

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics and Data Analysis*, **161**, 107245.

See Also

`chi2.test`, `chisq.test`

Examples

```
lambda <- get.lambda.function(N = 200, g = 50, model = 4)
Lmax <- get.lambda.max(N = 200, g = 50, model = 4)
X <- rstpoispp(lambda, Lmax)
result <- chisq.test.stPP(X, n.space = 2, n.time = 2)
print(result)
```

`dHS.test` *dHSIC test for first-order separability of a spatio-temporal point process*

Description

Performs a nonparametric test of first-order separability between space and time in a spatio-temporal point process using the distance-based Hilbert–Schmidt independence criterion (dHSIC). The test statistic evaluates whether the spatio-temporal intensity can be factorized as a function of spatial coordinate times as a function of temporal coordinate.

Usage

```
dHS.test(
  X,
  sim.procedure = c("pure_per", "block_per"),
  nblocks = 7L,
  nperm = 1999L,
  nsim = 199L,
  bandwidth = NULL
)
```

Arguments

`X`	A numeric matrix or data frame with at least three columns giving event coordinates (x, y, t).
`sim.procedure`	Character string specifying the permutation strategy: `"pure_per"` or `"block_per"`.
`nblocks`	Integer (>= 2). Number of temporal blocks for block permutation (only for `"block_per"`).
`nperm`	Integer (>= 1). Number of block permutations (only for `"block_per"`).
`nsim`	Integer (>= 1). Number of pure permutations (only for `"pure_per"`).
`bandwidth`	Optional numeric. Fixed bandwidth to use with `kernel = "gaussian.fixed"` in `dHSIC::dhsic`. If provided, the function also returns a fixed-bandwidth Monte Carlo p-value. If `NULL`, only the adaptive-bandwidth Gaussian kernel (`kernel = "gaussian"`) is used.

Details

Two permutation strategies are supported:

`"pure_per"` Randomly permutes the time component across events.

`"block_per"` Uses block-wise permutation of the time component via `sim.procedures` with `method = "block"` to preserve short-range temporal dependence.

The Monte Carlo p-value is computed with the standard +1 correction: $(1+\#\{T_i \geq T_{obs}\})/(B+1)$, where B is the number of permutations.

Value

A list with components:

p.value Monte Carlo p-value based on the adaptive-bandwidth Gaussian kernel.

p.value.bw Monte Carlo p-value based on the fixed-bandwidth Gaussian kernel, or `NA` if `bandwidth = NULL`.

bandwidth_data Bandwidth selected by `dHSIC::dhsic(..., kernel = "gaussian")`.

Note

The dHSIC method is implemented via the **dHSIC** package. When `sim.procedure = "pure_per"`, `dHS.test()` internally calls `global.envelope.test` for computational efficiency.

References

Ghorbani, M., Vafaei, N. and Myllymäki, M. (2025). A kernel-based test for the first-order separability of spatio-temporal point processes, *TEST*.

See Also

`sim.procedures`, `block.permut`, `chi2.test`

Examples

```
if (requireNamespace("dHSIC", quietly = TRUE)) {
  set.seed(123)
  X <- cbind(runif(100), runif(100), runif(100, 0, 10))

  # Pure permutation test
  result <- dHS.test(sim.procedure = "pure_per",
                     X = X, nsim = 199, bandwidth = 0.05)
  print(result$p.value)

  # Block permutation test
  result_block <- dHS.test(sim.procedure = "block_per", X = X,
                           nblocks = 5, nperm = 100, bandwidth = 0.05)
  print(result_block$p.value.bw)
}
```

`estimate.intensity.pixel` *Kernel-based intensity estimation on a space-time grid and its components, and test statistics for first-order separability*

Description

Computes kernel-smoothed estimates of spatial, temporal, separable, and non-separable spatio-temporal intensity functions on a regular space-time grid, together with separability diagnostics used in first-order separability analysis.

Usage

```
estimate.intensity.pixel(X, s.region, t.region, n.grid, edge, owin = NULL)
```

Arguments

`X`	Numeric matrix/data.frame with three columns `(x, y, t)` giving observed events.
`s.region`	Numeric matrix with two columns defining the spatial window (typically polygon vertices). Grid limits are taken as `range(s.region[,1])` and `range(s.region[,2])`.
`t.region`	Numeric vector of length 2 giving the temporal window `c(tmin, tmax)`.
`n.grid`	Integer vector of length 3 giving grid resolution in x, y, and t.
`edge`	List with components `bw` (length 3), `space` (length `nrow(X)`), and `time` (length `nrow(X)`).
`owin`	Optional observation window of class `"owin"` (from **spatstat.geom**). If provided, intensity estimates outside the window are set to `NA`.

Details

The estimator uses product Gaussian kernels with supplied bandwidths and (Gaussian) edge-correction factors, typically produced by `calc.bandwidths.and.edgecorr`.

Value

A list with grid coordinates `x,y,t`, intensity estimates, the diagnostic `S.fun`, its marginal summaries `S.space` and `S.time`, and deviation measures.

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, 161, 107245.

See Also

`S.based.functions`, `calc.bandwidths.and.edgecorr`

Examples

```
n <- 100
X <- cbind(x = stats::runif(n), y = stats::runif(n), t = stats::runif(n, 0, 10))
s.region <- matrix(c(0,0, 1,0, 1,1, 0,1), ncol=2, byrow=TRUE)
t.region <- c(0, 10)
n.grid <- c(10, 10, 5)
edge <- list(bw = c(0.05, 0.05, 0.5), space = rep(1, n), time = rep(1, n))
res <- estimate.intensity.pixel(X, s.region, t.region, n.grid, edge)
str(res)
```

`estimate.intensity.point`

Kernel intensity estimates of a spatio-temporal point process at observed points and its components, and test statistics for first-order separability

---

Description

Computes kernel-based spatial, temporal, separable, and non-separable intensity estimates evaluated at the observed spatio-temporal event locations. The function also returns the separability diagnostic S_i and global deviation measures quantifying departures from first-order separability.

Usage

```
estimate.intensity.point(X, n.grid, edge)
```

Arguments

`X`	Numeric matrix/data.frame with three columns `(x,y,t)` giving event coordinates.
`n.grid`	Integer. Included for API compatibility with grid-based routines; not used.
`edge`	List with components `bw` (length 3), `space`, and `time`. `space` and `time` are Gaussian edge-correction masses evaluated at each event; each may be a scalar or a numeric vector of length `nrow(X)`.

Details

Pairwise Gaussian kernel weights are computed in each dimension and diagonal entries are set to zero to remove self-contributions.

Value

A list with components `S.fun`, deviation measures, and estimated intensity components at the observed points.

See Also

dnorm

Examples

```
X <- cbind(stats::runif(50), stats::runif(50), stats::runif(50))
edge <- list(bw = c(0.1, 0.1, 0.1), space = 1, time = 1)
res <- estimate.intensity.point(X, n.grid = 50, edge = edge)
str(res)
```

---

estimate.st.intensity *Kernel Estimation of the Spatio-Temporal Intensity Function and Its Components, and Test Statistics for First-Order Separability*

Description

It returns estimates of the full spatio-temporal intensity together with its spatial and temporal marginal components. The output also includes the test statistic $S(u, t)$ and its spatial and temporal marginal profiles $S_{\text{space}}(u)$ and $S_{\text{time}}(t)$ (equations (11)–(13) in Ghorbani et al., 2021), as well as several deviation tests (e.g., equation (15) in Ghorbani et al., 2021) that quantify departures from first-order separability.

Usage

```
estimate.st.intensity(
  X,
  s.region,
  t.region,
  at = c("pixels", "points"),
  n.grid = c(25, 25, 20)
)
```

Arguments

X	A numeric matrix with three columns giving the event coordinates (x, y, and t).
s.region	A numeric matrix with two columns defining the polygonal boundary of the spatial observation region.
t.region	A numeric vector of length 2 specifying the temporal observation window.
at	Character string; either "pixels" to estimate intensity on a spatio-temporal grid or "points" to compute the estimates at observed event locations.
n.grid	A numeric vector of length 3 giving the number of grid cells in the x, y, and t dimensions. Required when at = "pixels".

Details

The estimation follows the kernel framework of Ghorbani et al. (2021). A Gaussian kernel is applied in each spatial and temporal dimension. Spatial bandwidths are estimated using Diggle's method, and the temporal bandwidth is obtained by the Sheather–Jones direct plug-in (SJ-DPI) approach (see `calc.bandwidths.and.edgecorr` for implementation details). Edge corrections are computed analytically using Gaussian tail probabilities (see `calc.bandwidths.and.edgecorr` for implementation details).

The non-separable intensity estimate is

$$\hat{\rho}(u,t) = \sum_{i=1}^{n} \frac{K^2_{\epsilon}(u-u_i)}{C_{W,\epsilon}(u_i)} \frac{K^1_{\delta}(t-t_i)}{C_{T,\delta}(t_i)},$$

where $k_b(v) = k(v/b)/b^d$, and K^1 and K^2 are Gaussian kernels with spatial and temporal bandwidths ϵ and δ. $C_{W,\epsilon}(u_i)$ and $C_{T,\delta}(t_i)$ are spatial and temporal edge corrections, respectively. For estimate of the separable counterparts and more details see equations (3)-(5) in Ghorbani et al., 2021).

The separability test is:

$$S(u,t) = \frac{\hat{\rho}(u,t)}{\hat{\rho}_{\text{space}}(u)\hat{\rho}_{\text{time}}(t)/n},$$

where n is the number of observed points.

Several deviation statistics are provided to quantify departures from separability

- `deviation.t1`: Integral of absolute deviations $\int_{W\times T} |\hat{\rho}(u,t) - \hat{\rho}_{\text{sep}}(u,t)| \mathrm{d}\, u \mathrm{d}\, t$.
- `deviation.t2`: Integral of absolute deviation of inverse intensities $\int_{W\times T} |1/\hat{\rho}(u,t) - 1/\hat{\rho}_{\text{sep}}(u,t)| \mathrm{d}\, u \mathrm{d}\, t$.
- `deviation.t3`: Sum of log-ratio deviations $\sum_i \big(\log(\hat{\rho}(u_i,t_i) - \log(\hat{\rho}_{\text{sep}}(u_i,t_i))\big)$.
- `deviation.t4`: Integral of the S-function.

When `at = "points"`, intensities and diagnostics are evaluated at the observed event locations.

Value

A list containing:

x, y, t Grid vectors for x, y, and t (returned only when `at = "pixels"`).

epsilon Estimated spatial bandwidth used in Gaussian kernels.

delta Estimated temporal bandwidth.

SPat.intens Estimated spatial intensity surface (pixels only).

TeM.intens Estimated temporal intensity profile (pixels only).

sep.intens Separable spatio-temporal intensity estimate.

nonsep.intens Non-separable spatio-temporal intensity estimate.

S.fun Test function $S(u,t)$ as the ratio of non-separable to separable intensity estimates.

S.space Marginal sum of $S(u,t)$ over time (space profile).

S.time Marginal sum of $S(u,t)$ over space (time profile).

deviation.t1 Deviation statistic: Integral of absolute deviations.

deviation.t2 Deviation statistic: absolute deviation of inverse intensities.

deviation.t3 Deviation statistic: sum of log-ratio deviations.

deviation.t4 Total integral of the S-function.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@ltu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

calc.bandwidths.and.edgecorr, global.envelope.test, chi2.test, dHS.test

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), runif(100, 0, 10))
s.region <- matrix(c(0, 0, 1, 0, 1, 1, 0, 1), ncol = 2, byrow = TRUE)
t.region <- c(0, 10)
result <- estimate.st.intensity(X, s.region, t.region, at = "pixels", n.grid = c(64, 64, 32))
str(result$S.fun)
image(result$S.fun[,,5], main = "S-function slice at t[5]",
       col = topo.colors(50))
contour(result$S.fun[,,5], add=TRUE)
str(result[c("deviation.t1","deviation.t3","deviation.t4")])
```

Gauss.st.F *Simulate a spatio-temporal Gaussian random field on a regular grid*

Description

Simulates a space–time Gaussian random field on a regular (x, y, t) grid. The field is returned as a 3D array and can be used as a latent field for log-Gaussian Cox process (LGCP) simulation.

Usage

```
Gauss.st.F(
  xlim = c(0, 1),
  ylim = c(0, 1),
  tlim = c(0, 1),
  par1 = c(1, 0.05),
```

```
  par2 = c(1, 0.06),
  sigmas = c(0.5, 0.5, 1),
  grid = c(15L, 15L, 10L)
)
```

Arguments

`xlim`, `ylim`, `tlim`	Numeric vectors of length 2 giving the ranges for the spatial and temporal axes. Defaults are `c(0,1)` for each.
`par1`	Numeric vector of length 2 giving the temporal covariance parameters `c(variance, scale)` for an exponential covariance $var * exp(-d/scale)$.
`par2`	Numeric vector of length 2 giving the spatial covariance parameters `c(variance, scale)` for an exponential covariance $var * exp(-d/scale)$.
`sigmas`	Numeric vector of length 3 specifying the weights $(\sigma_1, \sigma_2, \sigma_3)$ for combining the spatial, temporal, and spatio-temporal components of the field.
`grid`	Integer vector of length 3 giving the number of grid points in the x, y, and t directions.

Details

The simulated field is a weighted sum of three independent Gaussian components:

$$Z(x, y, t) = \sigma_1 Z_s(x, y) + \sigma_2 Z_t(t) + \sigma_3 Z_{st}(x, y, t),$$

where Z_s is a purely spatial field, Z_t is a purely temporal field, and Z_{st} is a spatio-temporal field with separable exponential covariance in space and time.

The function uses `mvrnorm` for multivariate normal simulation and `rdist` to compute pairwise distances for covariance matrix construction.

Spatial and temporal covariances are exponential. The spatio-temporal component uses a separable covariance $C_{st}((u, t), (u', t')) = C_s(u, u')C_t(t, t')$. Simulation is performed via Cholesky factors without constructing the full $(nx * ny * nt) \times (nx * ny * nt)$ covariance matrix.

Value

A list with components:

Z Numeric array of dimension `c(nx, ny, nt)` containing simulated field values.

xcoord Numeric vector of length `nx` with x-grid coordinates.

ycoord Numeric vector of length `ny` with y-grid coordinates.

tcoord Numeric vector of length `nt` with time-grid coordinates.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics and Data Analysis*, **161**, 107245.

See Also

mvrnorm, rdist

Examples

```
if (requireNamespace("MASS", quietly = TRUE) && requireNamespace("fields", quietly = TRUE)) {
  set.seed(1)
  field <- Gauss.st.F(
    xlim = c(0, 1), ylim = c(0, 1), tlim = c(0, 1),
    par1 = c(1, 0.05), par2 = c(1, 0.06),
    sigmas = c(0.5, 0.5, 1),
    grid = c(15, 15, 10)
  )
# Inspect dimensions and visualize one time slice
dim(field$Z)
image(field$xcoord, field$ycoord, field$Z[, , 1],
      main = "Gaussian Random Field (t = 1)",
      col = RColorBrewer::brewer.pal(11, "Spectral"))
  }
```

`get.lambda.function` *Construct spatio-temporal intensity functions with controlled separability*

Description

Returns an intensity function $\lambda(x, y, t)$ corresponding to one of four models used for simulation experiments in Ghorbani et al. (2021). Each model is a mixture of a separable "background" component and a structured (generally non-separable) spatio-temporal Gaussian bump. The models provide different degrees of space–time separability, allowing for controlled experiments on separability testing.

Usage

```
get.lambda.function(
  N,
  g,
  model = 1L,
  mu1 = 0.5,
```

```
  sd1 = 0.2,
  mu2 = c(0.5, 0.5),
  sd2 = c(0.2, 0.2),
  mu3 = c(0.3, 0.3, 0.2),
  sd3 = c(0.05, 0.05, 0.05)
)
```

Arguments

N	Numeric scalar (> 0). Baseline intensity level (interpreted as an expected total count after scaling in the calling simulator; see details below).
g	Numeric scalar (>= 0). Weight of the structured (non-separable) component.
model	Integer in 1 : 4 indicating the structure of the intensity function.
mu1	Numeric scalar. Mean of the temporal Gaussian background term (models 2 and 4).
sd1	Numeric scalar (> 0). Standard deviation of the temporal Gaussian background term (models 2 and 4).
mu2	Numeric vector of length 2. Mean of the spatial 2D Gaussian background term (models 3 and 4).
sd2	Numeric vector of length 2 with positive entries. Standard deviations of the spatial 2D Gaussian background term (models 3 and 4).
mu3	Numeric vector of length 3. Mean of the structured (non-separable) 3D Gaussian component.
sd3	Numeric vector of length 3 with positive entries. Standard deviations of the structured 3D Gaussian component.

Details

The returned function is intended for use in simulation (e.g., for generating spatio-temporal Poisson point patterns under varying degrees of separability).

The intensity is constructed as:

$$\lambda(x, y, t) = \lambda_{\text{bg}}(x, y, t) + g\, f_{st}(x, y, t),$$

where f_{st} is a nonnegative 3D Gaussian density (via norm3d) and the background term λ_{bg} depends on model:

1 Homogeneous background: $\lambda_{\text{bg}}(x, y, t) = (N - g)$

2 Temporal inhomogeneity only: $\lambda_{\text{bg}}(x, y, t) = (N - g)\, f_t(t)$, where f_t is a 1D Gaussian density (dnorm).

3 Spatial inhomogeneity only: $\lambda_{\text{bg}}(x, y, t) = (N - g)\, f_s(x, y)$, where f_s is a 2D Gaussian density (norm2d).

4 Separable spatial-temporal inhomogeneity: $\lambda_{\text{bg}}(x, y, t) = (N - g)\, f_s(x, y)\, f_t(t)$.

Note: since Gaussian densities can exceed 1 for small standard deviations, N is best interpreted as a scaling parameter used by the calling simulator. Ensure $\lambda(x, y, t)$ is nonnegative over the intended domain.

See more details in Ghorbani et al. (2021), Section 6.1.

Value

A function of the form `function(x, y, t)` representing the selected intensity surface.

Note

This function is primarily intended for generating intensity functions used in simulation studies. In particular, `rstpoispp` calls `get.lambda.function()` internally to construct intensity models for simulating spatio-temporal Poisson point processes with controlled separability.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

`rstpoispp`, `norm3d`, `norm2d`, `chi2.test`, `dHS.test`

Examples

```
# Choose model 4: non-separable spatio-temporal intensity
lambda <- get.lambda.function(N = 210, g = 50, model = 4)
lambda(0.5, 0.5, 0.5)  # Evaluate intensity at center of space-time domain

# Visualize spatial intensity at fixed time for model 2
lambda2 <- get.lambda.function(N = 200, g = 50, model = 2)
x <- y <- seq(0, 1, length.out = 100)
z <- outer(x, y, function(x, y) lambda2(x, y, t = 0.5))
par(mar = c(5, 4, 4, 6))
fields::image.plot(x, y, z, main = "Intensity at t = 0.5 (Model 2)", col = topo.colors(50))
```

---

get.lambda.max *Upper bound for spatio-temporal intensity models*

Description

Computes a practical upper bound for the spatio-temporal intensity models used in `get.lambda.function`. The bound is intended for thinning/rejection sampling in simulation routines such as `rstpoispp`.

Usage

```
get.lambda.max(
  N,
  g,
  model = 1L,
  mu1 = 0.5,
  sd1 = 0.2,
  mu2 = c(0.5, 0.5),
  sd2 = c(0.2, 0.2),
  mu3 = c(0.3, 0.3, 0.2),
  sd3 = c(0.05, 0.05, 0.05)
)
```

Arguments

N	Numeric scalar (> 0). Total expected number of events (baseline intensity level).
g	Numeric scalar (>= 0). Weight of the structured (non-separable) component.
model	Integer in 1:4. See get.lambda.function.
mu1	Numeric scalar. Mean of the temporal Gaussian background term (models 2 and 4).
sd1	Numeric scalar (> 0). Standard deviation of the temporal Gaussian background term.
mu2	Numeric vector of length 2. Mean of the spatial Gaussian background term (models 3 and 4).
sd2	Numeric vector of length 2 with positive entries. Standard deviations of the spatial background term.
mu3	Numeric vector of length 3. Mean of the structured spatio-temporal Gaussian component.
sd3	Numeric vector of length 3 with positive entries. Standard deviations of the structured component.

Details

The bound is computed using analytic maxima of Gaussian density components (at their modes), which yields a conservative and fast-to-evaluate upper bound when the component functions are Gaussian product densities.

The intensity models are mixtures of a background term and a structured spatio-temporal Gaussian bump. This function returns an upper bound obtained by evaluating each Gaussian density component at its mode. This upper bound is typically sufficient for rejection sampling when generating realizations of inhomogeneous Poisson or Cox point processes.

If norm2d and norm3d in this package are Gaussian product densities (independent components), then the maxima are available in closed form:

- $\max_t \phi(t; \mu, \sigma) = 1/(\sigma\sqrt{2\pi})$
- $\max_{x,y} \phi(x; \mu_x, \sigma_x)\phi(y; \mu_y, \sigma_y) = 1/(2\pi\sigma_x\sigma_y)$

- $\max_{x,y,t} \prod_{d=1}^{3} \phi(\cdot; \mu_d, \sigma_d) = 1/((2\pi)^{3/2}\sigma_x\sigma_y\sigma_t)$

If your `norm2d`/`norm3d` use a different parameterization, this bound should be updated accordingly.

Value

Numeric scalar. A conservative upper bound for the selected intensity model.

Author(s)

Nafiseh Vafaei <nafiseh.vafaei@slu.se>
Mohammad Ghorbani <mohammad.ghorbani@slu.se>

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics and Data Analysis*, **161**, 107245.

See Also

`get.lambda.function`, `rstpoispp`

Examples

```
# Example 1: Homogeneous model (Model 1)
get.lambda.max(N = 200, g = 50, model = 1)

# Example 2: Non-separable spatio-temporal model (Model 4)
get.lambda.max(N = 200, g = 50, model = 4)
```

---

global.envelope.test *Global envelope test for spatio-temporal separability using S-function*

Description

Performs a global envelope test of the null hypothesis of first-order separability for a spatio-temporal point process. The observed separability diagnostics $S(u,t)$, $S_{\text{space}}(u)$ and/or $S_{\text{time}}(t)$ are compared to a reference distribution obtained from permuted versions of the data.

Usage

```
global.envelope.test(
  X,
  sim.procedure = c("pure_per", "block_per"),
  nsim = 25L,
  nblocks = 5L,
  nperm = 199L,
```

```
  n.grid = c(20L, 20L, 10L),
  s.region = matrix(c(0, 0, 1, 0, 1, 1, 0, 1), ncol = 2, byrow = TRUE),
  t.region = c(0, 1),
  owin = NULL,
  eps = NULL,
  del = NULL,
  tests = c("S.test", "S.space.test", "S.time.test"),
  GET.args = NULL
)
```

Arguments

X	A numeric matrix or data frame with at least three columns giving (x, y, t).
sim.procedure	Character string specifying the permutation strategy: "pure_per" or "block_per".
nsim	Integer. Number of permutations for "pure_per".
nblocks	Integer (>= 2). Number of temporal blocks for block permutation. Used only for "block_per".
nperm	Integer. Number of block permutations for "block_per".
n.grid	Integer/numeric vector of length 3 specifying grid resolution in x, y, t.
s.region	Numeric matrix with two columns specifying polygon vertices of the spatial window.
t.region	Numeric vector of length 2 specifying temporal window c(tmin,tmax).
owin	Optional window of class "owin" (from **spatstat.geom**). If supplied, values outside the window are set to NA (pixels mode).
eps	Optional numeric scalar (>0). Spatial bandwidth. If NULL, estimated internally.
del	Optional numeric scalar (>0). Temporal bandwidth. If NULL, estimated internally.
tests	Character vector indicating which diagnostics to test. Any of "S.test", "S.space.test", "S.time.test".
GET.args	Optional named list of extra arguments passed to global_envelope_test (e.g. alternative, savefuns, nstep).

Details

Two permutation strategies are supported:

"pure_per" Pure permutation: randomly permutes the time coordinates.

"block_per" Block permutation: permutes time in blocks to preserve short-range temporal dependence.

The **GET** package is used to construct global envelopes and compute p-values.

The null hypothesis is

$$H_0 : \ \rho(u,t) = \rho_{\text{space}}(u)\rho_{\text{time}}(t).$$

The function computes the chosen diagnostics using S.based.functions on a pixel grid (at="pixels") and applies global_envelope_test.

To keep curve lengths identical (required by **GET**), any NA values induced by an owin mask are removed using the **same indices** for the observed and all simulated curves.

Value

A list with components:

Bandwidth_s Spatial bandwidth used.

Bandwidth_t Temporal bandwidth used.

S.test, S.space.test, S.time.test For each requested test: p-values for ERL and AREA envelopes, plus optional plots if created.

Author(s)

Nafiseh Vafaei <nafiseh.vafaei@slu.se>
Mohammad Ghorbani <mohammad.ghorbani@slu.se>

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

S.based.functions, sim.procedures, block.permut, global_envelope_test, plot.global_envelope

Examples

```
if (requireNamespace("GET", quietly = TRUE)) {
  set.seed(123)
  X <- cbind(stats::runif(100), stats::runif(100), stats::runif(100, 0, 1))
  s.region <- matrix(c(0,0, 1,0, 1,1, 0,1), ncol = 2, byrow = TRUE)
  t.region <- c(0, 1)

  res <- global.envelope.test(
    X = X,
    sim.procedure = "pure_per",
    nsim = 19,
    n.grid = c(10,10,10),
    s.region = s.region,
    t.region = t.region,
    tests = c("S.test","S.time.test")
  )
  str(res)
}
```

norm2d *Bivariate normal density with independent components*

Description

Evaluates the density of a bivariate normal distribution with mean vector $\mu = (\mu_x, \mu_y)$ and diagonal covariance matrix (independent components). The density is the product of the two univariate normal densities:

$$f(x, y) = \phi(x; \mu_x, \sigma_x)\, \phi(y; \mu_y, \sigma_y).$$

Usage

```
norm2d(x, y, mu = c(0, 0), sd = c(1, 1), log = FALSE)
```

Arguments

x	Numeric vector of x-coordinate(s).
y	Numeric vector of y-coordinate(s).
mu	Numeric vector of length 2 giving c(mu_x, mu_y).
sd	Numeric vector of length 2 giving positive standard deviations c(sd_x, sd_y).
log	Logical; if TRUE, return the log-density.

Value

Numeric vector of densities (or log-densities) with length determined by standard recycling rules for x and y.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@slu.se>

Examples

```
# Evaluate the density at the peak
norm2d(0.5, 0.5, mu = c(0.5, 0.5), sd = c(0.2, 0.2))

# Evaluate at multiple x values
norm2d(c(0.3, 0.7), 0.5, mu = c(0.5, 0.5), sd = c(0.2, 0.2))

# Visualize on a grid
x <- y <- seq(0, 1, length.out = 100)
f <- Vectorize(function(x, y) norm2d(x, y, mu = c(0.5, 0.5), sd = c(0.2, 0.2)))
z <- outer(x, y, f)
image(x, y, z, col = terrain.colors(50), main = "Bivariate Normal Intensity")
contour(x, y, z, add = TRUE)
```

norm3d *Trivariate normal density with independent components in space-time*

Description

Evaluates a trivariate normal density on (x, y, t) with independent components (diagonal covariance). The density is the product of three univariate normal densities:

$$f(x, y, t) = \phi(x; \mu_x, \sigma_x)\, \phi(y; \mu_y, \sigma_y)\, \phi(t; \mu_t, \sigma_t).$$

Usage

```
norm3d(x, y, t, mu = c(0.3, 0.3, 0.2), sd = c(0.05, 0.05, 0.05), log = FALSE)
```

Arguments

x	Numeric vector of x-coordinate(s).
y	Numeric vector of y-coordinate(s).
t	Numeric vector of time coordinate(s).
mu	Numeric vector of length 3 giving c(mu_x, mu_y, mu_t).
sd	Numeric vector of length 3 giving positive standard deviations c(sd_x, sd_y, sd_t).
log	Logical; if TRUE, return the log-density.

Value

Numeric vector of densities (or log-densities) with length determined by standard recycling rules for x, y, and t.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

norm2d, get.lambda.function, estimate.st.intensity

Examples

```
norm3d(0.3, 0.3, 0.2)  # peak value at the mean (with default parameters)
norm3d(c(0.2, 0.3), 0.3, 0.2)


x <- y <- seq(0, 1, length.out = 100)
z <- outer(x, y, function(x, y) norm3d(x, y, t = 0.2))
image(x, y, z, col = heat.colors(50), main = "Spatial slice of norm3d at t = 0.2")
```

`plot_procedures` *Plot original vs permuted time ordering*

Description

Produces a side-by-side plot comparing the temporal component of an original spatio-temporal point pattern with that of a permuted (or block-permuted) version. This graphical diagnostic is intended to assess the effect of temporal reordering procedures used in separability tests.

Usage

```
plot_procedures(
  original,
  permuted,
  title = "Permutation",
  col = c("blue", "red"),
  pch = 1,
  ...
)
```

Arguments

`original`	A matrix or data frame with at least three columns (x, y, t). The time coordinate is taken from column 3.
`permuted`	A matrix or data frame with the same structure as `original`. The time coordinate is taken from column 3.
`title`	Character string; title for the permuted panel.
`col`	Character vector of length 2 giving colors for the original and permuted panels.
`pch`	Plotting character passed to `plot`.
`...`	Further graphical parameters passed to `plot`.

Details

The function is commonly employed to visualize temporal permutations generated by procedures such as `sim.procedures` or `block.permut`, which underpin pure and block permutations-based inference for first-order separability (see Ghorbani et al., 2021, Section 3.2).

The function uses base R graphics to display two panels:

1. The temporal ordering of the original point pattern.
2. The temporal ordering after permutation or block permutation.

This diagnostic is particularly useful when validating permutation-based inference procedures such as `chi2.test`, `dHS.test`, and `global.envelope.test`.

Value

The function is invoked for its side effect of producing a plot and returns no value.

Author(s)

Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani, M., Vafaei, N., Dvořák, J., & Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

`sim.procedures`, `block.permut`, `dHS.test`, `chi2.test`

Examples

```
set.seed(123)
X <- cbind(x = runif(20), y = runif(20),
           time = seq(0, 1, length.out = 20))

# Example: visualize pure permutation
sim_pure <- sim.procedures(X, nperm = 1, method = "pure")[[1]]
plot_procedures(X, sim_pure, title = "Pure Permutation")

# Example: visualize block permutation (requires Block_permutation)
if (requireNamespace("combinat", quietly = TRUE)) {
  sim_block <- block.permut(nblocks = 4, X = X, nperm = 1)[[1]]
  plot_procedures(X, sim_block, title = "Block Permutation")
}
```

plot_stDPP *Plot spatio-temporal determinantal point process (DPP) realizations*

Description

Produces diagnostic plots for spatio-temporal determinantal point process (DPP) simulations or fitted models. The function formats the plot title based on the spatial and temporal interaction parameters α_s and α_t, automatically displaying either the parameters themselves or their reciprocals when greater than 1.

Usage

```
plot_stDPP(data, type = c("3D", "space", "time"), alpha_s, alpha_t)
```

Arguments

data	A spatio-temporal point pattern object suitable for `plot_stpp()`, typically from the **stpp** or related packages.
type	Character string specifying the type of plot to produce. This is passed directly to `plot_stpp()`. Typical values are `"3D"`, `"space"`, and `"time"`.
alpha_s	Numeric scalar (> 0). Spatial interaction parameter of the DPP model.
alpha_t	Numeric scalar (> 0). Temporal interaction parameter of the DPP model.

Details

If $\alpha_s > 1$ and $\alpha_t > 1$, the title displays α_s^{-1} and α_t^{-1}, which correspond to interaction ranges. Otherwise, the parameters are shown directly.

The function then calls `plot.ST.pp()` to produce the actual plot.

Value

No return value. The function is called for its side effect of producing a diagnostic plot.

Examples

```
# Simulate a stationary separable Matérn ST-DPP
sim <- rstDPP(
  mode     = "stationary",
  model    = "S",
  spectral = "matern",
  alpha_s  = 10,
  alpha_t  = 4.7,
  nu       = 2,
  eps      = 1,
  lambda_max = 70,
  grid_size  = 1.5
)
```

```
plot_stDPP(sim, type = "3D",  alpha_s = 10, alpha_t = 4.7)
```

`plot_stlgcp` *Plot spatio-temporal log-Gaussian Cox process (LGCP) realizations as time-sliced maps*

Description

Produces a sequence of spatial raster maps displaying the evolution of a simulated or fitted spatio-temporal log-Gaussian Cox process (LGCP) over time. Each map shows the latent Gaussian random field (intensity surface) at a given time slice, with the corresponding observed point events overlaid. This visualization helps interpret temporal evolution and spatial clustering patterns in simulated or fitted LGCP models.

Usage

```
plot_stlgcp(data)
```

Arguments

`data`	A list containing components from a spatio-temporal LGCP model or simulation output: `RF` A list describing the latent Gaussian random field, with elements `xcoord`, `ycoord`, `tcoord`, and `Z`, where `Z` is a 3D array of intensity (or log-intensity) surfaces over space and time. `st.lgcp` A data frame with columns `x`, `y`, and `t` giving the spatial and temporal coordinates of observed events.

Details

The function plots up to 10 evenly spaced time slices from the latent intensity field and overlays the corresponding point events accumulated up to each time point. Each panel represents a spatial realization at a fixed time t_k, providing a visual summary of the dynamic spatio-temporal structure of the LGCP.

This approach follows the visualization principles used in Ghorbani et al. (2021, 2025), where LGCPs are employed to assess the behavior of separability diagnostics and kernel-based tests in complex, non-separable point process settings. The raster intensity surfaces illustrate latent heterogeneity, while the overlaid points display observed event clustering relative to the underlying field.

Time slices are selected at evenly spaced quantiles of the temporal domain, and each plot includes:

- A raster map of the latent intensity surface at time t_k;
- White contour lines showing equal-intensity regions;
- Overlaid black points representing events observed up to t_k.

The resulting plots are arranged into a single grid layout using the **patchwork** package for ease of comparison.

Value

A combined ggplot object displaying up to ten spatial raster maps arranged in a grid layout (by default, two rows and up to five columns). Each panel corresponds to one time slice. The function returns the combined plot object.

Author(s)

Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

Gauss.st.F for simulating spatio-temporal Gaussian random fields; get.lambda.function and rstpoispp for generating intensity-based spatio-temporal point processes.

Examples

```
# Example: visualize a spatio-temporal LGCP simulation
out <- rstLGCPP(xlim = c(0,1),
                ylim = c(0,1),
                tlim = c(0,1),
                grid = c(15,15,10))
plot_stlgcp(data = out)
```

---

plot_stpp *Plot a spatio-temporal point pattern*

Description

Provides flexible visualization tools for spatio-temporal point patterns. Depending on the chosen display type, the function produces one of three plots:

- A 3D scatterplot showing event locations in space–time ("3D");
- A 2D spatial projection of points in the spatial plane ("space");
- A temporal histogram with an overlaid kernel density curve ("time").

The input dataset must contain three columns corresponding to the spatial (x, y) and temporal (t) coordinates of events.

Usage

```
plot_stpp(data, type = c("3D", "space", "time"), time_bins = 30, title = NULL)
```

Arguments

`data`	A numeric matrix or data frame with at least three columns representing event coordinates. If `data` is a matrix, the first three columns are interpreted as `x`, `y`, and `t`. If `data` is a data frame, the function uses columns named `x`, `y`, `t` when present; otherwise it uses the first three columns.
`type`	Character string specifying the type of visualization to produce: `"3D"` for a three-dimensional scatterplot; `"space"` for a 2D spatial plot of `x` versus `y`; or `"time"` for a histogram of event times with a smoothed density overlay.
`time_bins`	Integer specifying the number of bins in the histogram when `type = "time"`. Default is 30.
`title`	Optional character string giving a plot title. If `NULL`, a default title is used.

Details

The function serves as an exploratory tool for investigating the spatial, temporal, or joint space–time structure of point pattern data. Such visualization is often a first step before conducting statistical analyses of separability or intensity modeling (see Ghorbani et al., 2021).

3D mode Displays events in a 3D coordinate system using the **scatterplot3d** package, allowing a quick assessment of clustering and temporal trends in space–time.

Spatial mode Projects points onto the spatial plane (x–y), showing spatial structure independent of time.

Temporal mode Displays the marginal temporal distribution of events as a histogram with a kernel density overlay, facilitating the visual detection of temporal nonstationarity.

Visualization is particularly useful for validating the realism of simulated data (e.g. from `rstpoispp` or `Gauss.st.F`) and for preliminary inspection prior to applying tests such as `chi2.test`, `global.envelope.test`, or `dHS.test`.

Value

Produces a plot as a side effect. Nothing is returned.

Note

The 3D visualization requires the **scatterplot3d** package.

Author(s)

Nafiseh Vafaei <nafiseh.vafaei@slu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

See Also

`rstpoispp`, `estimate.st.intensity`, `plot_stlgcp`, `chi2.test`, `dHS.test`

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), runif(100, 0, 10))

# Visualize point pattern in 3D space-time
plot_stpp(X, type = "3D")

# View spatial projection
plot_stpp(X, type = "space")

# Inspect temporal distribution
plot_stpp(X, type = "time", time_bins = 20)
```

`random.shift` *Apply a circular random shift to the temporal component of a spatio-temporal point pattern*

Description

Performs a circular random shift of the temporal coordinate in a spatio-temporal point pattern. This operation preserves the spatial configuration while randomizing the temporal component under the assumption of temporal stationarity. The shift amount is drawn uniformly from $[0, 1]$ and applied modulo 1, ensuring that the time window is maintained. The resulting dataset can be used to construct null models for hypothesis testing of first-order separability or temporal independence.

Usage

```
random.shift(X, shifted_col = 3)
```

Arguments

`X`	A numeric matrix or data frame containing the spatio-temporal point pattern. Must include at least one numeric column representing time.
`shifted_col`	Integer index specifying which column to shift (typically the time coordinate). Default is `3`.

Details

The circular random shift is a common resampling procedure for generating null models of temporal randomness while preserving the overall temporal marginal distribution and spatial structure.

For each dataset, a single uniform random shift value $\Delta \sim \mathrm{Uniform}(0, 1)$ is drawn and added to the temporal coordinate. The shifted times are then wrapped around the unit interval:

$$t_i^* = (t_i + \Delta) \bmod 1, \quad i = 1, \ldots, n.$$

Value

A matrix or data frame (matching the input type) with the time column shifted modulo 1.

References

Ghorbani, M., Vafaei, N., Dvořák, J., and Myllymäki, M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, 161, 107245.

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), runif(100))  # x, y, t
X_shifted <- random.shift(X)

# Compare original and shifted time values
head(X[,3])
head(X_shifted[,3])

# Verify shift visually
plot(X[, 3], type = "o", col = "blue", ylab = "Time", xlab = "Index",
     main = "Original vs Shifted Times")
lines(X_shifted[, 3], type = "o", col = "red")
legend("topright", legend = c("Original", "Shifted"),
       col = c("blue", "red"), lty = 1, pch = 1)
```

---

rstDPP *Simulate a spatio-temporal Determinantal Point Process (DPP) based on spectral density*

Description

Generates a realization of a spatio-temporal determinantal point process (DPP) using a user-defined spectral density model. The function supports both separable (`model = "S"`) and non-separable (`model = "NS"`) dependence structures and allows for either exponential or Mat\'ern-type spectral densities. The simulation is performed on a 3D spatio-temporal frequency grid and can incorporate user-specified intensity functions for thinning.

Usage

```
rstDPP(
  mode = c("stationary", "inhom"),
  model = c("S", "NS"),
  spectral = c("exp", "matern"),
  alpha_s,
  alpha_t = NULL,
```

```
  lambda_max,
  nu = 2,
  eps = 1,
  lambda_s = NULL,
  lambda_non_s = NULL,
  grid_size = 4
)
```

Arguments

mode	Character. Type of dependence model: "Stationary" A homogeneous spatio-temporal DPP is generated. "Inhomogeneous" A non-homogeneous spatio-temporal DPP is generated.
model	Character. Type of dependence model: "S" Separable spatio-temporal covarince function model, where space and time components are separable. "NS" Non-separable spatio-temporal model, allowing interaction between space and time.
spectral	Character. Type of spectral density function to use: "exp" for exponential spectral form or "matern" for a Mat\'ern-type spectral model.
alpha_s	Numeric. Spatial decay or range parameter in the spectral density.
alpha_t	Numeric. Temporal decay or range parameter in the spectral density.
lambda_max	Numeric. The maximum intensity value used for thinning. Must be specified.
nu	Numeric. Smoothness parameter for the Mat\'ern-type spectral density (only relevant if spectral = "matern"). Default is 2.
eps	Numeric. Degree of separability for the Mat\'ern model: eps = 0 corresponds to full non-separability, eps = 1 yields complete separability, and intermediate values provide partial separability.
lambda_s	Optional. Intensity function lambda_s(u, t) for separable models. If not provided, a default function is used.
lambda_non_s	Optional. Intensity function lambda_non_s(u, t) for non-separable models. If not provided, a default function is used.
grid_size	Numeric. Half-width of the spatio-temporal frequency grid. The total grid size is (2 * grid_size + 1)^3.

Details

This function implements a spectral simulation method for spatio-temporal DPPs, following the theoretical framework introduced in Vafaei et al. (2023) and Ghorbani et al. (2025).

The algorithm proceeds as follows:

1. Construct a 3D grid of spatial and temporal frequency components $(\omega_x, \omega_y, \tau)$.
2. Evaluate the chosen spectral density $\phi(\omega, \tau)$ across the grid.
3. Use the resulting spectral values as eigenvalues to simulate a realization of a DPP via $spatstat.model :: rdpp()$.

4. Optionally apply thinning using a user-defined intensity function $\lambda(u, t)$, scaled by `lambda_max`, to induce inhomogeneity.

Two spectral families are supported:

- **Exponential form:**

$$\phi(\omega, \tau) \propto \exp\left[-(\pi\alpha_s|\omega|)^2\right]\left(1 + 4(\pi\alpha_t\tau)^2\right)^{-1}.$$

- **Mat\'ern-type form:**

$$\phi_\epsilon(\omega, \tau) \propto \left(\alpha_s^2\alpha_t^2 + \alpha_t^2|\omega|^2 + \alpha_s^2\tau^2 + \epsilon|\omega|^2\tau^2\right)^{-\nu},$$

  where $\epsilon \in [0, 1]$ determines the degree of separability between space and time.

This framework enables simulation of spatio-temporal point patterns that exhibit varying degrees of spatial–temporal dependence, providing a versatile tool for evaluating separability tests and modeling non-separable dynamics.

Value

A numeric matrix with three columns (`x`, `y`, `t`) representing the retained spatio-temporal events after thinning.

References

Vafaei, N., Ghorbani, M., Ganji, M., and Myllymäki, M. (2023). Spatio-temporal determinantal point processes. *arXiv:2301.02353*.

Ghorbani, M., Vafaei, N., and Myllymäki, M. (2025). A kernel-based test for the first-order separability of spatio-temporal point processes. *TEST*, 34, 580-611. https://doi.org/10.1007/s11749-025-00972-y

See Also

`plot_stpp` for visualizing spatio-temporal point patterns.

Examples

```
# Simulate a stationary separable Mat\'ern ST-DPP
 if (requireNamespace("spatstat", quietly = TRUE)) {
sim <- rstDPP(
  mode     = "stationary",
  model    = "S",
  spectral = "matern",
  alpha_s  = 10,
  alpha_t  = 4.7,
  nu       = 2,
  eps      = 1,
  lambda_max = 70,
  grid_size  = 2
```

```
)
 plot_stDPP(sim, type = "3D",  alpha_s = 10, alpha_t = 4.7)
# example 2
# Generate realization
sim <- rstDPP(mode = "stationary",
               model = "S",
               spectral = "matern",
               alpha_s = 10, alpha_t = 4.7,
               nu = 2,
              eps = 1,
              lambda_s = 70,
              lambda_non_s = NULL,
              grid_size = 2,
              lambda_max=70)
head(sim)
}
```

`rstLGCPP` *Simulate a spatio-temporal Log-Gaussian Cox process (LGCP)*

Description

Generates a realization of a spatio-temporal LGCP over a user-defined domain. The process is simulated using a log-Gaussian random field combined with a deterministic trend function, and points are generated by thinning a homogeneous Poisson process.

Usage

```
rstLGCPP(
  xlim = NULL,
  ylim = NULL,
  tlim = NULL,
  grid = c(15, 15, 10),
  mu = NULL,
  Lambda = NULL,
  Lmax = NULL,
  par1 = c(1, 0.05),
  par2 = c(1, 0.06),
  sigmas = c(0.5, 0.5, 1),
  mu_par = c(1.2, 0.25, 5)
)
```

Arguments

`xlim`, `ylim`, `tlim`	Numeric vectors of length 2 specifying the spatial and temporal domains.
`grid`	Integer vector of length 3 specifying the number of grid cells in x, y, and t.

mu	Optional. A function of (x, y, t, par) defining a deterministic trend. Default is nonlinear.
Lambda	Optional. A user-supplied 3D intensity array or function. If NULL, it's generated from the latent Gaussian field.
Lmax	Optional. Maximum intensity used for thinning. Can be numeric or a function. If NULL, it's computed automatically.
par1, par2	Parameters for temporal and spatial exponential covariance models, respectively.
sigmas	Weights for combining spatial, temporal, and spatio-temporal components of the latent Gaussian field.
mu_par	Parameters passed to the default trend function mu() if not user-supplied.

Value

A list with:

st.lgcp A data frame of simulated spatio-temporal points.

RF The latent Gaussian field output from Gauss.st.F.

Examples

```
out <- rstLGCPP(xlim = c(0,1),
                ylim = c(0,1),
                tlim = c(0,1),
                grid = c(15,15,10))
plot_stlgcp(data = out)
plot_stpp(data = out$st.lgcp, type = "3D")
```

---

rstpoispp *Simulate an inhomogeneous spatio-temporal Poisson point process*

Description

Generates a realization of an inhomogeneous Poisson point process (STPP) in space and time using the standard thinning method. The user provides an intensity function $\lambda(u, t)$ and an upper bound $L_{\max}$ on its value over the observation window. The algorithm first samples candidate events uniformly over space and time and then retains each candidate with probability proportional to its normalized intensity $\lambda(u, t)/L_{\max}$.

Usage

```
rstpoispp(
  lambda,
  Lmax,
  s.region = splancs::as.points(c(0, 1, 1, 0), c(0, 0, 1, 1)),
  t.region = c(0, 1)
)
```

Arguments

`lambda`	A function of the form `lambda(u, t)` that returns the intensity value at coordinates `(x, y, t)`.
`Lmax`	A numeric value giving the known or estimated maximum of the intensity function `lambda` over the spatial and temporal window. Used for thinning.
`s.region`	A matrix with two columns giving the polygonal spatial window. Each row is a vertex of the polygon. Default is the unit square.
`t.region`	A numeric vector of length 2 giving the temporal observation window. Default is `c(0,1)`.

Details

The method implements the classical *thinning algorithm* for simulating inhomogeneous Poisson processes:

1. Draw $N^* \sim \text{Poisson}(L_{\max} |W| |T|)$, where $|W|$ and $|T|$ denote the spatial and temporal window measures.
2. Generate N^* candidate points uniformly over $W \times T$.
3. Retain each point (u_i, t_i) independently with probability $p_i = \lambda(u_i, t_i)/L_{\max}$.

The result is a realization of an inhomogeneous STPP with intensity function $\lambda(u, t)$.

This simulator underpins the spatio-temporal framework introduced in Ghorbani et al. (2021, 2025) for studying *first-order separability*. By selecting appropriate intensity functions (see `get.lambda.function`), users can generate fully separable, partially separable, or non-separable spatio-temporal patterns, enabling direct evaluation of separability tests such as `chi2.test`, `global.envelope.test`, or `dHS.test`.

Value

A numeric matrix with three columns (`x`, `y`, `t`) representing the retained points from the inhomogeneous Poisson process.

Note

The intensity function $\lambda(u, t)$ should return non-negative numeric values and be bounded above by `Lmax` across the observation domain.

Author(s)

Mohammad Ghorbani <mohammad.ghorbani@slu.se>
Nafiseh Vafaei <nafiseh.vafaei@ltu.se>

References

Ghorbani M., Vafaei N., Dvořák J., Myllymäki M. (2021). Testing the first-order separability hypothesis for spatio-temporal point patterns. *Computational Statistics & Data Analysis*, **161**, 107245.

Ghorbani, M., Vafaei, N. and Myllymäki, M. (2025). A kernel-based test for the first-order separability of spatio-temporal point processes, *TEST* .

See Also

get.lambda.function to construct spatio-temporal intensity models; get.lambda.max to compute intensity maxima; estimate.st.intensity for intensity estimation; plot_stpp for visualization.

Examples

```
# Example 1: Simulate a separable spatio-temporal Poisson process
lambda <- get.lambda.function(N = 200, g = 50, model = 1)
Lmax   <- get.lambda.max(N = 200, g = 50, model = 1)
X <- rstpoispp(lambda, Lmax)
head(X)

# Example 2: Non-separable model (Model 4)
lambda <- get.lambda.function(N = 200, g = 50, model = 4)
Lmax   <- get.lambda.max(N = 200, g = 50, model = 4)
sim_data <- rstpoispp(lambda, Lmax)

# Spatial projection of simulated events
plot(sim_data[, 1:2], asp = 1, main = "Spatial Projection of Simulated stPP")
# Example 3: 3D visualization using plot_ST_pp()
plot_stpp(X, type = "3D", title="Realisation of a stPP")
```

S.based.functions *Compute S-based test function for testing the null hypothesis of first-order separability*

Description

Computes kernel-based estimates of the spatio-temporal intensity and related separability diagnostics, either on a regular spatio-temporal grid ("pixels") or at the observed event locations ("points").

Usage

```
S.based.functions(
  X,
  s.region,
  t.region,
  owin = NULL,
  at = c("pixels", "points"),
  n.grid = c(25L, 25L, 20L),
  epsilon = NULL,
  delta = NULL,
  output = "all"
)
```

Arguments

X	Numeric matrix/data.frame with three columns giving (x, y, t).
s.region	Numeric matrix with two columns giving polygon vertices of the spatial window.
t.region	Numeric vector of length 2 giving the temporal window c(tmin, tmax).
owin	Optional spatial window of class "owin" (from **spatstat.geom**). Used only when at="pixels" to set values outside the window to NA.
at	Character string: "pixels" or "points".
n.grid	Integer vector of length 3 giving the grid resolution in x, y, and t (used when at="pixels"; still checked for length/positivity in both modes).
epsilon	Optional numeric scalar (>0). Spatial bandwidth. If NULL, estimated internally.
delta	Optional numeric scalar (>0). Temporal bandwidth. If NULL, estimated internally.
output	Character string selecting which component to return. Use "all" (default) to return all available components; otherwise return only the selected component along with bandwidths and coordinates.

Details

The function is a wrapper that (i) validates inputs, (ii) computes bandwidths and Gaussian edge-correction masses via calc.bandwidths.and.edgecorr, and (iii) delegates the actual estimation to estimate.intensity.pixel or estimate.intensity.point.

When at="points", spatial/temporal profiles such as S.space and S.time are typically not defined and are returned as NULL by the pointwise routine.

Value

If output="all", returns the full list produced by the chosen computation routine, augmented with epsilon and delta.

If output is a single component name, returns a list with:

S The requested component.

epsilon Spatial bandwidth used.

delta Temporal bandwidth used.

x,y,t Coordinates returned by the underlying routine (may be NULL).

Examples

```
X <- cbind(stats::runif(50), stats::runif(50), stats::runif(50))
s.region <- matrix(c(0,0, 1,0, 1,1, 0,1), ncol = 2, byrow = TRUE)
t.region <- c(0, 1)
res_all <- S.based.functions(X, s.region, t.region, at = "points")
res_Sfun <- S.based.functions(X, s.region, t.region, at = "points", output = "S.fun")
```

`sim.procedures` *Generate permuted versions (pure or block) of a spatio-temporal point pattern*

Description

Implements two types of permutation procedures for resampling the time component of spatio-temporal point process data:

`"pure"` Pure random permutation of the time coordinates.

`"block"` Block permutation where the time dimension is divided into consecutive blocks, and permutations are applied at the block level.

These procedures are used for generating surrogate datasets under the null hypothesis of first-order separability.

Usage

```
sim.procedures(X, nperm = 1999, nblocks = 4, method = c("block", "pure"))
```

Arguments

`X`	A numeric matrix or data frame with at least three columns, where the third column represents time.
`nperm`	Integer. The number of permuted datasets to generate.
`nblocks`	Integer. The number of temporal blocks to use for block permutation. Must be > 2.
`method`	Character. The permutation strategy to use. One of `"pure"` or `"block"`.

Value

A list of `nperm` matrices. Each matrix is a permuted version of the original input `X`, where the third column (time) has been resampled based on the selected method.

Examples

```
set.seed(123)
X <- cbind(runif(100), runif(100), sort(runif(100)))

# Pure permutation
sims_pure <- sim.procedures(X, nperm = 10, method = "pure")
head(sims_pure[[1]])
# Block permutation
sims_block <- sim.procedures(X, nperm = 10, nblocks = 5, method = "block")

# Visualize the first result from block permutation
plot_stpp(sims_block[[1]], type = "3D")
```

Index

block.permut, 2, *8*, *11*, *24*, *28*
bw.diggle, *4, 5*

calc.bandwidths.and.edgecorr, 4, *12*, *15, 16*, *41*
check.args, 5
chi2.test, 6, *9*, *11*, *16*, *20*, *28*, *32*, *39*
chisq.test, *9*
chisq.test.stPP, *7*, 8, *8*

density, *4, 5*
dHS.test, 10, *16*, *20*, *28*, *32*, *39*
dnorm, *14*, *19*

estimate.intensity.pixel, 12, *41*
estimate.intensity.point, 13, *41*
estimate.st.intensity, 14, *26*, *32*, *40*

Gauss.st.F, 16, *31, 32*, *38*
get.lambda.function, 18, *20–22*, *26*, *31*, *39, 40*
get.lambda.max, 20, *40*
global.envelope.test, *11*, *16*, 22, *28*, *32*, *39*
global_envelope_test, *23, 24*

mvrnorm, *17, 18*

norm2d, *19, 20*, 25, *26*
norm3d, *19, 20*, 26

owin, *5*

plot, *27*
plot.global_envelope, *24*
plot_procedures, 27
plot_stDPP, 29
plot_stlgcp, 30, *32*
plot_stpp, 31, *36*, *40*
ppp, *5*

random.shift, 33
rdist, *17, 18*
rstDPP, 34
rstLGCPP, 37
rstpoispp, *20*, *22*, *31, 32*, 38

S.based.functions, *12*, *23, 24*, 40
sim.procedures, *3*, *7, 8*, *11*, *24*, *28*, 42