Package 'BorderEffect'

August 5, 2026

Title Detection of Edge Effects in Field Trials via Besag-Kempton Competition

Version 0.1.0

Description Estimates and evaluates the intraspecific competition coefficient associated with the edge (border) effect in agricultural field trials, using the Besag-Kempton autoregressive model and a least-squares estimator following Darghan, Rivera, Gonzalez and Castellanos (2022) <doi:10.47280/RevFacAgron(LUZ).v39.n1.18>. Provides field-layout generation, spatial weight matrices, a formula interface for arbitrary two-way designs, a simulation-based decision procedure for presence or absence of the edge effect, Moran's I diagnostics, and a 'shiny' application.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, graphics, grDevices

Suggests shiny, ape, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Sergio Gamboa [aut, cre, cph] (Author of the R package),
Aquiles Enrique Darghan Contreras [aut, cph] (Author of the method and original code),
Carlos Armando Rivera Moreno [aut, cph] (Author of the method and original code),
Nair Jose Gonzalez Sotomayor [aut, cph] (Author of the method),
Jose Luis Castellanos Coronel [aut, cph] (Author of the method)

Maintainer Sergio Gamboa <segamboam@gmail.com>

Repository CRAN

Date/Publication 2026-08-05 06:30:16 UTC

Contents

`border_effect` *Estimate the edge-effect competition coefficient*

Description

Fits the least-squares Besag-Kempton competition coefficient κ for a two-way (treatment + block) field trial. The projection onto the design's column space is obtained from `model.matrix(formula, data)` via `qr.resid()`, so any design expressible as an R formula is supported.

Usage

```
border_effect(formula, data, coords, weights = NULL)
```

Arguments

`formula`	A model formula, e.g. `response ~ trt + blk`.
`data`	A data.frame containing the response and the design factors.
`coords`	A `field_layout`, or a data.frame/matrix of `x`/`y` coordinates aligned row-wise with `data`.
`weights`	Optional precomputed weight matrix (see `border_weights()`); built from `coords` when `NULL`.

Value

An object of class `border_effect`: a list with `kappa`, `call`, `formula`, `terms`, `weights`, `residuals`, `fitted`, `coords`, `data`.

Note

`field_layout()` reserves the column names `x` and `y` for plot coordinates. When your `data` is a `field_layout` (or shares its coordinates via `coords`), give the response a different name (e.g. `resp`, `yield`, `response`) so it does not overwrite the `y` coordinate.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 12, ny = 8, ntrt = 3, nblk = 2, seed = 3000)
set.seed(1); fl$resp <- rnorm(96, 2.21, 0.06)
be <- border_effect(resp ~ trt + blk, data = fl, coords = fl)
be
```

`border_indicators` *Edge (border) indicator vector from plot coordinates*

Description

Given the coordinates of the units in a rectangular field trial, returns a 0/1 indicator of the units belonging to a chosen border pattern. The outer ring (`"outer1"`) and the second ring (`"outer2"`) correspond to the sets S_1 and S_2 of the paper; the side and multi-side patterns reproduce the partial-border scenarios.

Usage

```
border_indicators(
  coords,
 pattern = c("outer1", "outer2", "lower", "left", "upper", "right", "lower_left",
    "lower_upper", "lower_right", "left_upper", "left_right", "upper_right",
  "lower_left_upper", "lower_left_right", "lower_upper_right", "left_upper_right",
    "central_h")
)
```

Arguments

`coords`	A `field_layout`, a data.frame with x/y, or a numeric matrix.
`pattern`	One of `"outer1"`, `"outer2"`, `"lower"`, `"left"`, `"upper"`, `"right"`, `"lower_left"`, `"lower_upper"`, `"lower_right"`, `"left_upper"`, `"left_right"`, `"upper_right"`, `"lower_left_upper"`, `"lower_left_right"`, `"lower_upper_right"`, `"left_upper_right"`, `"central_h"`.

Value

An integer vector of 0s and 1s, length `nrow(coords)`.

Note

The side and ring patterns are defined from the global minimum/maximum `x` and `y` coordinates (matching the original method), so they assume an aligned rectangular grid. With `arrangement = "triangular"` in `field_layout()`, the half-shifted even rows are not fully captured by the side patterns (e.g. `"left"`/`"right"` only flag the exact-min/max-x column, which excludes the shifted units in even rows). Users of triangular layouts should interpret side/ring patterns accordingly.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 5, ny = 4, ntrt = 2, nblk = 2)
table(border_indicators(fl, "outer1"))
```

`border_test` *Simulation-based decision for the presence of the edge effect*

Description

Implements the paper's decision procedure: an indicator model `response ~ S1` (or `response ~ S1 + S2` for the double outer edge) is fitted, new responses are generated with `simulate()`, and κ is recomputed for each simulated response using the two-way design. The resulting distribution yields a confidence interval; if it contains zero the edge effect is judged absent, otherwise present.

Usage

```
border_test(object, pattern = "outer1", nsim = 500, level = 0.95, seed = NULL)
```

Arguments

`object`	A `border_effect` object.
`pattern`	Border pattern to test (see `border_indicators()`). `"outer2"` uses the double-edge model `~ S1 + S2`; all others use `~ S1`.
`nsim`	Number of simulated responses (default 500).
`level`	Confidence level for the interval (default 0.95).
`seed`	Optional seed; RNG state is restored on exit.

Value

An object of class `border_test`: a list with `kappa_obs`, `kappa_sim`, `ci`, `level`, `contains_zero`, `decision`, `pattern`, `nsim`.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 12, ny = 8, ntrt = 3, nblk = 2, seed = 3000)
set.seed(1); fl$resp <- rnorm(96, 2.21, 0.06)
be <- border_effect(resp ~ trt + blk, data = fl, coords = fl)
border_test(be, pattern = "outer1", nsim = 200, seed = 42)
```

border_weights *Spatial weight matrix from plot coordinates*

Description

Builds an inverse-distance weight matrix `W` for the units of a field trial, as used by the Besag-Kempton competition model. The diagonal is zero (a unit does not compete with itself) and, by default, the matrix is standardized to sum to one.

Usage

```
border_weights(coords, method = c("idw"), power = 1, standardize = TRUE)
```

Arguments

`coords`	A `field_layout`, a data.frame with columns `x` and `y`, or a numeric matrix whose first two columns are the coordinates.
`method`	Weighting method. Currently only `"idw"` (inverse distance).
`power`	Positive exponent applied to the distance before inverting.
`standardize`	If `TRUE` (default) divide by the total sum so `sum(W) == 1`.

Value

A symmetric numeric matrix of class `border_weights` with zero diagonal.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 4, ny = 4, ntrt = 2, nblk = 2)
W <- border_weights(fl)
sum(W)
```

field_layout *Generate a rectangular field-trial layout with border indicators*

Description

Places `nx * ny` experimental units on a rectangular grid (square or triangular / "tres bolillos" arrangement), randomly assigns treatments, blocks and replicates, and appends 0/1 indicator columns for every border pattern (see `border_indicators()`).

Usage

```
field_layout(
  nx,
  ny,
  ntrt = 3,
  nblk = 2,
  arrangement = c("square", "triangular"),
  width = NULL,
  height = NULL,
  randomize = TRUE,
  seed = NULL
)

## S3 method for class 'field_layout'
plot(x, color_by = "trt", ...)
```

Arguments

nx, ny	Number of columns and rows of the grid.
ntrt, nblk	Number of treatments and blocks.
arrangement	"square" or "triangular" (even rows shifted by half a column spacing).
width, height	Physical extent of the grid. Defaults to nx - 1, ny - 1.
randomize	If TRUE (default) randomize the treatment/block assignment.
seed	Optional integer seed; RNG state is restored on exit.
x	A field_layout object.
color_by	Column used to color the points (default "trt").
...	Passed to plot().

Value

A data.frame of class field_layout with columns x, y, trt, blk, rep, and one integer column per border pattern.

Note

With arrangement = "triangular", the appended border-pattern columns (see border_indicators()) are still computed from the global min/max x/y, which assumes an aligned rectangular grid; the half-shifted even rows mean side patterns like "left"/"right" only capture the exact-min/max-x column. Interpret side/ring indicators with this in mind for triangular layouts.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 12, ny = 8, ntrt = 3, nblk = 2, seed = 3000)
head(fl)
```

field_trial *Example field trial with a single outer edge effect*

Description

A simulated staggered-planting trial (12 x 8 units, 3 treatments, 2 blocks) where the outer ring has a higher mean response than the interior, used to demonstrate edge-effect detection.

Usage

```
field_trial
```

Format

A data.frame with 96 rows and 6 columns:

x, y plot coordinates

trt treatment factor (T1-T3)

blk block factor (B1-B2)

rep replicate factor

response simulated yield

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

kappa_hat *Least-squares estimator of the Besag-Kempton competition coefficient*

Description

Computes the closed-form least-squares estimator of the competition coefficient κ associated with the edge effect, given a response vector, a spatial weight matrix, and a design. The perpendicular projection onto the column space of the design is applied implicitly via `qr.resid()`, so no explicit $n \times n$ projection matrix is formed and the estimator works for any (possibly rank-deficient) design.

Usage

```
kappa_hat(y, W, design)
```

Arguments

`y`	Numeric response vector of length `n`.
`W`	Symmetric `n x n` spatial weight matrix (see `border_weights()`).
`design`	An `n x p` model matrix (e.g. from `model.matrix()`).

Value

A single numeric value, $\hat{\kappa}$.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
n <- 12
xy <- expand.grid(x = 1:4, y = 1:3)
W <- border_weights(xy)
X <- model.matrix(~ rep(0:1, length.out = n))
kappa_hat(rnorm(n), W, X)
```

`moran_border` *Moran's I spatial autocorrelation of model residuals*

Description

Computes Moran's I for the residuals of the fitted two-way model held in a `border_effect` object, using the same weight matrix used to estimate κ. A small p-value indicates spatial dependence of the residuals, a signature of the edge effect. The statistic and its normal-approximation variance follow the standard definition (as in `ape::Moran.I`).

Usage

```
moran_border(object)
```

Arguments

`object`	A `border_effect` object.

Value

A list with `observed`, `expected`, `sd`, and (two-sided) `p.value`.

References

Darghan, Rivera, Gonzalez & Castellanos (2022). doi:10.47280/RevFacAgron(LUZ).v39.n1.18

Examples

```
fl <- field_layout(nx = 6, ny = 4, ntrt = 4, nblk = 3, seed = 5)
set.seed(2); fl$resp <- rnorm(24)
be <- border_effect(resp ~ trt + blk, data = fl, coords = fl)
moran_border(be)
```

`run_border_app` *Launch the BorderEffect Shiny application*

Description

Opens an interactive application with two modes: **Analyze** a real trial (upload a CSV with `x`, `y`, `trt`, `block`, `response`, or pick one of the bundled example scenarios – no border, single/double outer border, and unilateral/bilateral/trilateral partial borders – then read the κ density, confidence interval and verdict) and **Explore** the method (adjust layout and effect size and watch the κ distribution respond). All statistics are computed by the package's exported functions.

Usage

```
run_border_app()
```

Value

A Shiny app object (invisibly, when run interactively).

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  run_border_app()
}
```

Index