Package 'smile'

May 9, 2026

Title Spatial Misalignment: Interpolation, Linkage, and Estimation

Version 1.1.0

Date 2025-09-11

Description Provides functions to estimate, predict and interpolate areal data. For estimation and prediction we assume areal data is an average of an underlying continuous spatial process as in Moraga et al. (2017) <doi:10.1016/j.spasta.2017.04.006>, Johnson et al. (2020) <doi:10.1186/s12942-020-00200-w>, and Wilson and Wakefield (2020) <doi:10.1093/biostatistics/kxy041>. The interpolation methodology is (mostly) based on Goodchild and Lam (1980, ISSN:01652273).

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

SystemRequirements GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)

LinkingTo Rcpp, RcppEigen

Imports numDeriv, Rcpp, sf, mvtnorm, stats, parallel, Matrix

Depends R (>= 4.1.0)

URL https://lcgodoy.me/smile/, https://github.com/lcgodoy/smile/

BugReports https://github.com/lcgodoy/smile/issues/

Suggests knitr, rmarkdown, ggplot2, graphics

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Author Lucas da Cunha Godoy [aut, cre] (ORCID: <https://orcid.org/0000-0003-4265-972X>)

Maintainer Lucas da Cunha Godoy <lcgodoy@duck.com>

Repository CRAN

Date/Publication 2025-09-22 22:00:02 UTC

Contents

`AI` *Areal Interpolation*

Description

This function estimates variables observed at a "source" region into a "target" region. "Source" and "target" regions represent two different ways to divide a city, for example. For more details, see https://lcgodoy.me/smile/articles/sai.html.

Usage

```
ai(source, target, vars)

ai_var(source, target, vars, vars_var, sc_vars = FALSE, var_method = "CS")
```

Arguments

`source`	a `sf` object - source spatial data.
`target`	a `sf` object - target spatial data.
`vars`	a `character` representing the variables (observed at the source) to be estimated at the target data.
`vars_var`	a scalar of type `character` representing the name of the variable in the source dataset that stores the variances of the variable to be estimated at the target data.
`sc_vars`	boolean indicating whether `vars` should be scaled by its observed variance (if available).
`var_method`	a `character` representing the method to approximate the variance of the AI estimates. Possible values are "CS" (Cauchy-Schwartz) or "MI" (Moran's I).

Value

the target (of type sf) with estimates of the variables observed at the source data.

---

find_phi *Find phi parameter for the Exponential spatial auto-correlation function*

Description

Function designed to find the phi parameter such that the correlation between points within a given distance d is at most a given value.

Usage

```
find_phi(
  d,
  nu,
  kappa,
  mu2,
  family = "matern",
  range = c(1e-04, 1000),
  cut = 0.05
)
```

Arguments

d	maximum distance for spatial dependence equal to cut.
nu	smoothness parameter associated with the Matern cov. function.
kappa	one of the smoothness parameters associated with the Generalized Wendland covariance function
mu2	one of the smoothness parameters associated with the Generalized Wendland covariance function
family	covariance function family, the options are c("matern","gw", "cs", "spher", "pexp", "gaussian").
range	Minimum and maximum distance to be considered. The default is range = c(1e-04, 1000).
cut	desired spatial correlation at a distance d, the default is cut = .05.

Value

a numeric value indicating the range parameter such that the spatial correlation between two points at distance d is cut.

`fit_spm` *Fitting an underlying continuous process to areal data*

Description

This function fits a spatial process model to areal data by estimating the parameters of an underlying continuous process. It leverages the stats::optim function for Maximum Likelihood estimation, providing flexibility in optimization algorithms and control parameters.

Usage

```
fit_spm(x, ...)

## S3 method for class 'spm'
fit_spm(
  x,
  model,
  theta_st,
  nu = NULL,
  kappa = 1,
  mu2 = 1.5,
  apply_exp = FALSE,
  opt_method = "Nelder-Mead",
  control_opt = list(),
  comp_hess = TRUE,
  ...
)

fit_spm2(
  x,
  model,
  nu,
  kappa = 1,
  mu2 = 1.5,
  comp_hess = TRUE,
  phi_min,
  phi_max,
  nphi = 10,
  cores = getOption("mc.cores", 1L)
)
```

Arguments

`x`	An object of type `spm`. The dimension of `theta_st` depends on the number of variables analyzed and whether the input is an `spm` object.
`...`	Additional parameters passed to stats::optim.

model	A character scalar indicating the covariance function family. Options are c("matern", "pexp", "gaussian", "spherical", "gw").
theta_st	A named numeric vector containing initial parameter values.
nu	A numeric value specifying the ν parameter for the Matern or Powered Exponential covariance functions. If model is "matern" and nu is not provided, it defaults to 0.5. If model is "pexp" and nu is not provided, it defaults to 1. In both cases, this results in the exponential covariance function.
kappa	$\kappa \in \{0, \ldots, 3\}$ parameter for the GW covariance function.
mu2	The smoothness parameter μ for the GW function.
apply_exp	A logical scalar indicating whether to exponentiate non-negative parameters.
opt_method	A character scalar specifying the optimization algorithm (see stats::optim).
control_opt	A named list of control arguments for the optimization algorithm (see stats::optim).
comp_hess	A logical indicating whether to compute the Hessian matrix.
phi_min	A numeric scalar representing the minimum phi value for grid search.
phi_max	A numeric scalar representing the maximum phi value for grid search.
nphi	A numeric scalar indicating the number of phi values for grid search.
cores	An integer scalar indicating the number of cores to use. Defaults to getOption("mc.cores"). No effect on Windows.

Details

The function utilizes optimization algorithms from stats::optim to determine Maximum Likelihood estimators (MLEs) and their standard errors for a model adapted for areal data. Users can customize the optimization process by providing control parameters through the control_opt argument (a named list, see stats::optim for details), specifying lower and upper bounds for parameters, and selecting the desired optimization algorithm via opt_method (must be available in stats::optim).

Additionally, the function supports various covariance functions, including Matern, Exponential, Powered Exponential, Gaussian, and Spherical. The apply_exp argument, a logical scalar, allows for exponentiation of non-negative parameters, enabling optimization over the entire real line.

The underlying model assumes a point-level process:

$$Y(\mathbf{s}) = \mu + S(\mathbf{s})$$

where $S\ GP(0, \sigma^2 C(\|\mathbf{s} - \mathbf{s}_2\|; \theta))$. The observed areal data is then assumed to behave as the average of the process over each area:

$$Y(B) = |B|^{-1} \int_B Y(\mathbf{s})\, \mathbf{ds}$$

.

Value

An spm_fit object containing the estimated model parameters.

Examples

```
data(liv_lsoa) ## loading the LSOA data

msoa_spm <- sf_to_spm(sf_obj = liv_msoa, n_pts = 500,
                      type = "regular", by_polygon = FALSE,
                      poly_ids = "msoa11cd",
                      var_ids = "leb_est")
## fitting model
theta_st_msoa <- c("phi" = 1) # initial value for the range parameter

fit_msoa <-
  fit_spm(x = msoa_spm,
          theta_st = theta_st_msoa,
          model = "matern",
          nu = .5,
          apply_exp  = TRUE,
          opt_method = "L-BFGS-B",
          control    = list(maxit = 500))

AIC(fit_msoa)

summary_spm_fit(fit_msoa, sig = .05)
```

`goodness_of_fit` *Akaike's (and Bayesian) An Information Criterion for* `spm_fit` *objects.*

Description

Akaike's (and Bayesian) An Information Criterion for `spm_fit` objects.

Usage

```
## S3 method for class 'spm_fit'
AIC(object, ..., k = 2)

## S3 method for class 'spm_fit'
BIC(object, ...)
```

Arguments

`object`	a `spm_fit` object.
`...`	optionally more fitted model objects.
`k`	`numeric`, the *penalty* per parameter to be used; the default 'k = 2' is the classical AIC. (for compatibility with `stats::AIC`.

Value

a `numeric` scalar corresponding to the goodness of fit measure.

`liv_lsoa` *Liverpool Lower Super Output Area.*

Description

A dataset containing containing the LSOA's for Liverpool along with estimates for Index of Multiple Deprivation. Data taken from Johnson et al. 2020

Usage

```
liv_lsoa
```

Format

A `sf` data frame with 298 rows and 6 variables:

lsoa11cd LSOA code

lsoa11cd LSOA name

male Male population

female Female population

imdscore Index of Multiple Deprivation

area LMSOA area, in km^2

Details

The data was projected to EPSG 27700 and units changed to km

Source

https://ij-healthgeographics.biomedcentral.com/articles/10.1186/s12942-020-00200-w

liv_msoa *Liverpool Middle Super Output Area.*

Description

A dataset containing containing the MSOA's for Liverpool along with estimates for Life Expectancy at Birth. Data taken from Johnson et al. 2020

Usage

```
liv_msoa
```

Format

A `sf` data frame with 61 rows and 4 variables:

msoa11cd MSOA code

msoa11cd MSOA name

lev_est Estimated life expectancy at birth, in years

area MSOA area, in km^2

Details

The data was projected to EPSG 27700 and units changed to km

Source

https://ij-healthgeographics.biomedcentral.com/articles/10.1186/s12942-020-00200-w

nl_ct *Nova Lima census tracts*

Description

A dataset containing containing the census tracts for the city of Nova Lima in Minas Gerais - Brazil.

Usage

```
nl_ct
```

Format

A sf data frame with 113 rows and 14 variables:

cd_setor unique identifier

hh_density average household density

var_hhd variance of the household density

avg_income average income per household

var_income variance of the income per household

pop population in the census tract

avg_age average age of the inhabitants in the census tract

var_age variance of the variable age in the census tract

prop_women proportion of women

prop_elder proportion of people with 55 years of age or older

illit_rate illiteracy rate

prop_white proportion of self-declared white people

prop_black proportion of self-declared black people

prop_native proportion of self-declared native people

Details

The data is project using the SIRGAS 2000.

---

predict_spm *Prediction over the same or a different set of regions (or points).*

Description

Realizes predictions that can be useful when researchers are interested in predict a variable observed in one political division of a city (or state) on another division of the same region.

Usage

```
predict_spm(x, ...)

## S3 method for class 'spm_fit'
predict_spm(x, .aggregate = TRUE, ...)

## S3 method for class 'sf'
predict_spm(x, spm_obj, n_pts, type, outer_poly = NULL, id_var, ...)
```

Arguments

x	a sf object such that its geometries are either points or polygons.
...	additional parameters
.aggregate	logical. Should the predictions be aggregated? In case the input is only a "fit" object, the aggregation is made over the polygons on which the original data was observed. In case the input x is composed by sf POLYGONS, the aggregation is made over this new partition of the study region.
spm_obj	an object of either class spm_fit or mspm_fit
n_pts	a numeric scalar standing for number of points to form a grid over the whole region to make the predictions
type	character type of grid to be generated. See st_sample in the package sf.
outer_poly	(object) sf geometry storing the "outer map" we want to compute the predictions in.
id_var	if x is a set of POLYGONS (areal data) instead of a set of points, the id_var is the name (or index) of the unique identifier associated to each polygon.

Value

a list of size 4 belonging to the class spm_pred. This list contains the predicted values and the mean and covariance matrix associated with the conditional distribution used to compute the predictions.

Examples

```
data(liv_lsoa) ## loading the LSOA data
data(liv_msoa) ## loading the MSOA data

msoa_spm <- sf_to_spm(sf_obj = liv_msoa, n_pts = 500,
                      type = "regular", by_polygon = FALSE,
                      poly_ids = "msoa11cd",
                      var_ids = "leb_est")
## fitting model
theta_st_msoa <- c("phi" = 1) # initial value for the range parameter

fit_msoa <-
   fit_spm(x = msoa_spm,
           theta_st = theta_st_msoa,
           model = "matern",
           nu = .5,
           apply_exp  = TRUE,
           opt_method = "L-BFGS-B",
           control    = list(maxit = 500))

pred_lsoa <- predict_spm(x = liv_lsoa, spm_obj = fit_msoa, id_var = "lsoa11cd")
```

sev_pexp *Calculate Smallest Eigenvalue for Power Exponential Correlation Matrices*

Description

This function computes the smallest eigenvalue of a correlation matrix derived from the power exponential correlation function. It evaluates this across a grid of values for the power parameter (nu) and the practical range parameter (rho), based on a provided distance matrix.

Usage

```
sev_pexp(range_nu, range_rho, grid_len = 50, dmat)
```

Arguments

range_nu	A numeric vector of length 2, specifying the minimum and maximum values for the power parameter nu. nu typically ranges between 0 and 2 (e.g., nu = 1 for exponential, nu = 2 for Gaussian).
range_rho	A numeric vector of length 2, specifying the minimum and maximum values for the practical range parameter rho. rho must be positive.
grid_len	An integer specifying the number of points to create for both nu and rho sequences. The total number of grid combinations will be grid_len^2. Default is 50.
dmat	A numeric matrix representing the distance matrix between locations. The distances should be non-negative.

Details

The practical range rho is defined here as the distance at which the correlation is 0.1. The internal scale parameter phi is calculated as phi = rho / (log(10)^(1/nu)). The power exponential correlation function is assumed to be of the form C(h) = exp(-(h/phi)^nu), where h is distance. The function smile:::pexp_cov is used internally to compute the covariance/correlation matrix with a sill of 1.

The function first creates a grid of nu and rho parameters. For each pair of (rho, nu) in the grid: 1. It calculates the scale parameter phi for the power exponential correlation function, where phi = rho / (log(10)^(1/nu)). This definition implies that the correlation is 0.1 at the distance rho. 2. It computes the power exponential correlation matrix using smile:::pexp_cov(dists = dmat, sill = 1, range = phi, smooth = nu). Note the use of an internal function from the smile package. 3. It calculates the eigenvalues of this correlation matrix. 4. The minimum eigenvalue is extracted. The final output is a tibble containing all parameter combinations and their corresponding minimum eigenvalues.

Value

A tibble with three columns:

rho	The practical range parameter value.
nu	The power parameter value.
lambda	The smallest eigenvalue of the power exponential correlation matrix corresponding to the rho and nu pair.

---

sf_to_spm *single* sf *to* spm

Description

Transforming a sf into a spm object (Internal use)

Usage

```
single_sf_to_spm(
  sf_obj,
  n_pts,
  type = "regular",
  by_polygon = FALSE,
  poly_ids = NULL,
  var_ids = NULL
)

sf_to_spm(
  sf_obj,
  n_pts,
  type = "regular",
  by_polygon = FALSE,
  poly_ids = NULL,
  var_ids = NULL
)
```

Arguments

sf_obj	a sf object s.t. its geometries are polygons.
n_pts	a numeric scalar representing the number of points to create a grid in the study region on which the polygons in sf_obj is observed. Alternatively, it can be a vector of the same length as nrow(sf_obj). In this case, it generates the given number of points for each polygon in sf_obj.
type	a character indicating the type of grid to be generated. The options are c("random", "regular", "hexagonal"). For more details, see st_sample in the sf package.

by_polygon	a logical indicating whether we should generate n_pts by polygon or for the n_pts for the whole study region.
poly_ids	a character vector informing the name of the variable in sf_obj that represents the polygons unique identifiers. In case this is not informed, we assume the id of the polygons are given by their row numbers.
var_ids	a scalar or vector of type character indicating the (numerical) variables that are going to be analyzed.

Value

a named list of size 6 belonging to the class spm. This list stores all the objects necessary to fit models using the fit_spm.

Examples

```
data(liv_lsoa) # loading the LSOA data

msoa_spm <- sf_to_spm(sf_obj = liv_msoa, n_pts = 1000,
                      type = "regular", by_polygon = FALSE,
                      poly_ids = "msoa11cd",
                      var_ids = "leb_est")
```

---

smile *smile: Spatial MIsaLignment Estimation*

Description

smile: Spatial MIsaLignment Estimation

---

summary_spm_fit *Summarizing* spm_fit

Description

Provides a data.frame with point estimates and confidence intervals for the parameters of the model fitted using the spm_fit function.

Usage

```
summary_spm_fit(x, sig = 0.05)
```

Arguments

x	a spm_fit object.
sig	a real number between 0 and 1 indicating significance level to be used to compute the confidence intervals for the parameter estimates.

Value

a data.frame summarizing the parameters estimated by the fit_spm function.

---

vdl *Voronoi Data Linkage*

Description

Reminder, have to create an example. This will be exported after we submit the paper for publication.

Usage

```
vdl(coords_sf, areal_sf, vars, buff)
```

Arguments

coords_sf	sf POINT target dataset.
areal_sf	sf POLYGON source dataset.
vars	a character representing the variables (observed at the source - polygon) to be estimated at the target data.
buff	scalar numeric. Mostly for internal use.

Value

a sf object for the coords_sf spatial data set.

---

vdl_var *Voronoi Data Linkage - Single variable and variance*

Description

Reminder, have to create an example. This will be exported after we submit the paper for publication.

Usage

```
vdl_var(coords_sf, areal_sf, res_var, variance, var_method = "CS", buff)
```

Arguments

coords_sf	sf POINT target dataset.
areal_sf	sf POLYGON source dataset.
res_var	a character - the name of the variable in the areal_sf to be estimated in the coords_sf.
variance	a character - the name of the variable variance in the areal_sf to be estimated in the coords_sf.
var_method	a character representing the method to approximate the variance of the AI estimates. Possible values are "CS" (Cauchy-Schwartz) or "MI" (Moran's I).
buff	scalar numeric. Mostly for internal use.

Value

a sf object, containing the id_coords variable and the list_vars for the coords_sf spatial data set.

Index