Package ‘smile’

October 14, 2022

Title  Spatial Misalignment: Interpolation, Linkage, and Estimation
Version  1.0.4.1
Date  2022-04-27
License  GPL-3
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.1
SystemRequirements  C++11, GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)
LinkingTo  Rcpp, RcppArmadillo
Imports  numDeriv, Rcpp, sf, mvtnorm, stats, Matrix
Depends  R (>= 3.5)
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]
  (<https://orcid.org/0000-0003-4265-972X>)
Maintainer  Lucas da Cunha Godoy <lucasdac.godoy@gmail.com>
Repository  CRAN
Date/Publication  2022-04-29 08:00:02 UTC
R topics documented:

AI ................................................................................. 2
find_phi ........................................................................... 3
fit_spm .............................................................................. 4
goodness_of_fit ................................................................. 7
liv_lsoa .............................................................................. 7
liv_msoa ............................................................................. 8
nyc_comd .......................................................................... 9
nyc_surv ........................................................................... 9
predict_spm ....................................................................... 10
sf_to_spm ......................................................................... 11
summary_spm_fit .............................................................. 12
vdl .................................................................................... 13
vdl_var ............................................................................. 14

Index 15

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).
find_phi

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

Examples

data(nyc_surv)
data(nyc_comd)

## creating variables that store the variance for each area
## this is done to exemplify the functionality of the package
nyc_surv <- transform(nyc_surv,  
                      my_var = moe / qnorm(p = .975))
nyc_surv <- transform(nyc_surv, my_var = my_var * my_var)

## Not run:
## areal interpolation
estimate_comd <-
  ai(source = nyc_surv, target = nyc_comd,  
      vars = "estimate")

## areal interpolation with uncertainty estimation
estimate_comd <-
  ai_var(source = nyc_surv, target = nyc_comd,  
         vars = "estimate", vars_var = "my_var",  
         var_method = "MI")

## End(Not run)

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

Fitting an underlying continuous process to areal data

Usage

```r
fit_spm(x, ...) 
```

## S3 method for class 'spm'

```r
fit_spm(
x, 
model, 
theta_st, 
u = NULL, 
tr = NULL, 
kappa = 1, 
mu2 = 1.5, 
apply_exp = FALSE, 
opt_method = "Nelder-Mead", 
control_opt = list(), 
comp_hess = TRUE, 
... 
)
```
fit_spm

```
fit_spm2(
  x,
  model,
  nu,
  tr,
  kappa = 1,
  mu2 = 1.5,
  comp_hess = TRUE,
  phi_min,
  phi_max,
  nphi = 10
)
```

**Arguments**

- **x**: an object of type `spm`. Note that, the dimension of `theta_st` depends on the 2 factors: 1) the number of variables being analyzed, and 2) if the input is a `spm` object.

- **...**: additional parameters, either passed to `optim`.

- **model**: a character scalar indicating the family of the covariance function to be used. The options are `c("matern", "pexp", "gaussian", "spherical", "gw")`.

- **theta_st**: a numeric (named) vector containing the initial parameters.

- **nu**: a numeric value indicating either the $\nu$ parameter from the Matern covariance function (controlling the process differentiability), or the "pexp" for the Powered Exponential family. If the `model` chosen by the user is Matern and `nu` is not informed, it is automatically set to .5. On the other hand, if the user choses the Powered Exponential family and do not inform `nu`, then it is set to 1. In both cases, the covariance function becomes the so-called exponential covariance function.

- **tr**: tapper range

- **kappa**: $\kappa \in \{0, \ldots, 3\}$ parameter for the GW (O) function.

- **mu2**: the smoothness parameter $\mu$ for the GW function.

- **apply_exp**: a logical scalar indicating whether the parameters that cannot assume negative values should be exponentiated or not.

- **opt_method**: a character scalar indicating the optimization algorithm to be used. For details, see `optim`.

- **control_opt**: a named list containing the control arguments for the optimization algorithm to be used. For details, see `optim`.

- **comp_hess**: a boolean indicating whether the Hessian matrix should be computed.

- **phi_min**: a numeric scalar representing the minimum `phi` value to look for.

- **phi_max**: a numeric scalar representing the maximum `phi` value to look for.

- **nphi**: a numeric scalar indicating the number of values to compute a grid-search over `phi`.
Details

This function uses the `optim` function optimization algorithms to find the Maximum Likelihood estimators, and their standard errors, from a model adapted from. The function allows the user to input the control parameters from the `optim` function through the argument `control_opt`, which is a named list. Additionally, the one can input lower and upper boundaries for the optimization problem, as well as the preferred optimization algorithm (as long as it is available for `optim`). The preferred algorithm is selected by the argument `opt_method`. In addition to the control of the optimization, the user can select a covariance function among the following: Matern, Exponential, Powered Exponential, Gaussian, and Spherical. The parameter `apply_exp` is a logical scalar such that, if set to TRUE, the `exp` function is applied to the nonnegative parameters, allowing the optimization algorithm to search for all the parameters over the real numbers.

The model assumes $Y(s) = \mu + S(s)$ at the point level. Where $S \sim GP(\theta, \sigma^2 C(\|s - s_2\|; \theta))$. Further, the observed data is supposed to be $Y(B) = \int_B Y(s) \, ds$.

Value

A `spm_fit` object containing the information about the estimation of the 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

```r
## 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

```r
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
- imdscor Index of Multiple Deprivation
- area LMSOA area, in $km^2$

Details

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

Source


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

**nyc_comd**

New York City community districts spatial geometries

**Description**

A dataset containing containing the CD’s for New York City.

**Usage**

nyc_comd

**Format**

A sf data frame with 71 rows and 3 variables:

- **boro_cd**: unique identifier
- **shape_area**: Shape Area
- **shape_length**: Shape Length
- **est**: median income estimated using areal interpolation
- **se_est**: standard error associated with the estimates

**Details**

The data is project using EPSG 4326.

---

**nyc_surv**

New York City survey data.

**Description**

A dataset containing containing the census tracts for New York City along with estimates for median income and a margin of error for this estimates.

**Usage**

nyc_surv

**Format**

A sf data frame with 2128 rows and 5 variables:

- **GEOID**: unique identifier
- **NAME**: census tract name
- **variable**: variable estimated
- **estimate**: median income estimate
- **moe**: median income estimate margin of error
Details

The data is project using EPSG 4326.

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 geometris are either points or polygons.
...
.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")

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,
 trunc_d = NULL
 )

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

trunc_d  

truncation distance for grid points. Consider using half of the maximum distance between polygons.

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")

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.
### vdl

**Usage**

```r
calliste_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 summarising 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**

```r
calliste(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.
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

\texttt{vd़l\_var(coords\_sf, areal\_sf, res\_var, variance, var\_method = \"CS\", buff)}

Arguments

- \texttt{coords\_sf} sf POINT target dataset.
- \texttt{areal\_sf} sf POLYGON source dataset.
- \texttt{res\_var} a character - the name of the variable in the \texttt{areal\_sf} to be estimated in the \texttt{coords\_sf}.
- \texttt{variance} a character - the name of the variable variance in the \texttt{areal\_sf} to be estimated in the \texttt{coords\_sf}.
- \texttt{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).
- \texttt{buff} scalar numeric. Mostly for internal use.

Value

a sf object, containing the \texttt{id\_coords} variable and the \texttt{list\_vars} for the \texttt{coords\_sf} spatial data set.
Index

* datasets
  liv_lsoa, 7
  liv_msoa, 8
  nyc_comd, 9
  nyc_surv, 9

AI, 2
ai (AI), 2
ai_var (AI), 2
AIC.spm_fit (goodness_of_fit), 7
BIC.spm_fit (goodness_of_fit), 7
find_phi, 3
fit_spm, 4, 12
fit_spm2 (fit_spm), 4
goodness_of_fit, 7
liv_lsoa, 7
liv_msoa, 8
nyc_comd, 9
nyc_surv, 9
predict_spm, 10
sf_to_spm, 11
single_sf_to_spm (sf_to_spm), 11
summary_spm_fit, 12

vd1, 13
vd1_var, 14